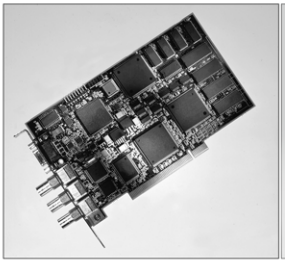
CineView® Pro XL
CineView® Pro
CineView® Pro LE

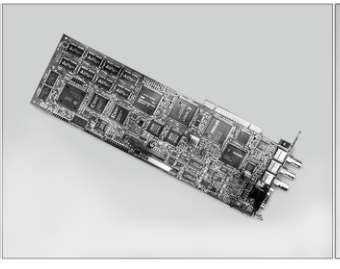
Vela's PCI MPEG-2 Decoder Family

Installation and User Manual

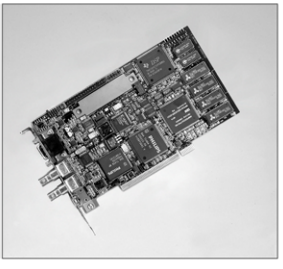
Includes a guide to the Vela Application Programming Interface Ver. 2.6 for the CineView Pro family of decoders



CineView® Pro



CineView® Pro XL



CineView® ProLE

Model 2000-2153
Model 2000-2100
Model 2000-2110

MPEG-2 Decoders for the PCI Bus

Release 2.6

Document Part Number 9050-2100

Vela OEM Products Division
5733 Myerlake Circle
Clearwater, FL 33760
Phone: (727) 507-5300
Fax: (727) 507-5310

Vela is a member of the PCI Special Interest Group, Vendor ID No. 0X127D

World Wide Web – http://www.vela.com

Mailing/Shipping Address: 5733 Myerlake Circle, Clearwater, FL 33760-2804

All returns must be accompanied by an authorized RMA number obtained from Vela.

FCC Information

This product has been tested and found to comply with FCC Rules Part 15 and meets all FCC Class B requirements.

CE

Printed in the United States of America 4-02

Rev. CP-0204-016

Table of Contents

List of Figures and Tables

Part One — Installation and Use

Chapter 1
Getting Started. 3

Part One

Installation and Use

Chapter 1

Getting Started

Introduction

Welcome to Vela's CineView® Pro family of audio/video decoders, version 2.6. These decoders include the model 2000-2153 CineView Pro XL the model 2000-2100 CineView Pro, and the model 2000-2110 CineView Pro LE. These broadcast-quality decoders are designed to support playback of MPEG-1 and MPEG-2 video clips on both the host computer's VGA monitor and/or an NTSC or PAL video system. Except as needed to distinguish unique board features or specifications, the decoders making up the CineView Pro family will be collectively referred to as "CineView Pro" in this publication.

The Vela CineView Pro family of decoders requires a Pentium®-powered personal computer with PCI bus architecture running Microsoft® Windows® 2000 or Windows NT™ 4.0 (with NT Service Pack 6a).

Document Organization

This manual is in three parts. Part One covers installation and general operation of the CineView Pro Family of decoders.

Part Two is a developer's guide to the most current Vela CineView Pro Application Programming Interface (API). An associated Software Developer's Kit (SDK) is available from Vela for those wishing to develop their own custom applications. Note that the CineView Pro API and SDK covers the CineView Pro XL as well as the CineView Pro and Pro LE. Contact your regional Vela sales representative for information on obtaining the CineView Pro SDK.

Part Three of this manual consists of two appendices and an index.

The CineView Pro Family

CineView Pro XL

The Vela CineView Pro XL decoder is designed to play back MPEG-1 and

NOTE: All trademarks, brand names, or product names appearing in this publication are registered to the respective companies or organizations that own the names or trademarks. "CineView" is a registered trademark of Vela LP for a line of digital audio/video decoding products.

MPEG-2 main-level main-profile system, transport, and program streams. Digital and composite video are output through respective BNC connectors on the rear of the decoder board. Composite video can also be played back through the system VGA monitor for convenient monitoring. An additional BNC on the rear of the decoder allows a genlock signal to be applied.

The CineView Pro XL decoder can output both serial D-1 video and NTSC/PAL composite video. Stream parsing is performed on board. Full-featured audio output capabilities are standard on the Pro XL. Both analog (+4dBm = 0Vu) and digital (AES/EBU) audio is output via a high-density 15-pin D-sub connector. In addition, SMPTE 337M AES/EBU compressed bitstream output is supported. S/P DIF audio can be output if desired by making the appropriate interface to a header connector located on the decoder. Digital audio can also be embedded in the SDI stream. Longitudinal Time Code is available at a header connector on the Pro XL decoder board and can be output through an optional add-on LTC connector bracket.

The CineView Pro XL is built on a full-length PCI form-factor board.

CineView Pro

The CineView Pro's SMPTE 259M and AES/EBU outputs allow simultaneous analog and digital output for both video and stereo audio. Video output is enabled through rugged, industry-standard BNC connectors. Audio output is provided via a high-density 15-pin connector. Stream parsing is performed right on board the CineView Pro family of decoders.

The CineView Pro (and Pro LE) are built on half-length PCI form factor boards.

CineView Pro LE

The CineView Pro LE decoder performs all the same functions as the Pro, but in an analog-only manner. The Pro LE does not support digital video or digital audio output. Stream parsing is performed on board the CineView Pro LE decoder.

CineView Pro Features

New for Release 2.6

Frame-Controlled Playback Interface: CineView Pro has a new SDK interface to allow for frame-accurate starts and stops. The interface also allows pausing on the first or last frame of video. You may specify a particular video frame or the corresponding time code to begin and/or end file playback.

Features Introduced in Release 2.5 and Earlier:

(XL Only) Enhanced Audio Capabilities: In addition to the audio capabilities found in the CineView Pro family of decoders, the CineView Pro XL supports four channels of audio embedded in the SDI stream. S/P DIF audio can be output through an optional connector bracket.

(XL Only) Longitudinal Time Code (LTC) Output: The Pro XL can output time code through an optional connector bracket.

Multiboard Playback Capability: Up to four CineView Pro decoders may be installed and run concurrently within a single PC system, depending on system configuration. Individual playback windows will appear for each decoder in use.

On Screen Display (of logos, etc.): OSD allows the user to place a bitmap image, such as a network or corporate logo, in the output video, at any position on the screen. The bitmap can be blended into the video with one of four levels of opacity to give a “see-through” look, or can be placed within a clear background. The bitmap can also be animated to present an appearance of motion.

Pause on First Frame: When Pause on First Frame is selected in the General Setup Window, Pause and Freeze Frame modes will display a frame of video. When unchecked (default), a field of video, rather than a frame, is displayed.

COM Functionality: The CineView Pro and Pro LE includes COM support for the CineView Pro API. This allows applications written in languages other than C++ to easily interface with the CineView Pro API and the decoder hardware. The CVProServer COM object wraps around the CineView Pro DLL containing all of the CineView Pro library functions, to present a standard binary interface for connection with customer developed application programs. Operation of the COM server should appear transparent to the user, once the application is implemented as a COM client. As of the CineView Pro SDK release 2.5, the CineView Pro application has been converted into a COM object called ProPlaybackClient.

GOP Time Code Support: CineView Pro and Pro LE versions 2.3 and forward include support to read and display GOP time codes from the MPEG GOP header. This dramatically improves accuracy of the displayed SMPTE time code compared with previous version 1.2, which displayed an approximate frame count using counters on board the Philips SAA7146A device. The DSP processor on the CineView Pro decoder board parses the MPEG data and reads the time codes from the GOP header. Unlike previous versions, the displayed GOP time code is accurate even during trick modes such as slow motion, fast forward, or pause.

Microcode Versioning Support: The 78-character versioning array from the

microcode file is read and displayed in the Help About window (Figure 1-1). Data displayed includes 4:2:2 and 4:2:0 microcode versions for the IBM decoder, the audio microcode version for the IBM decoder, and the build number, compile date and compile time for the DSP code.

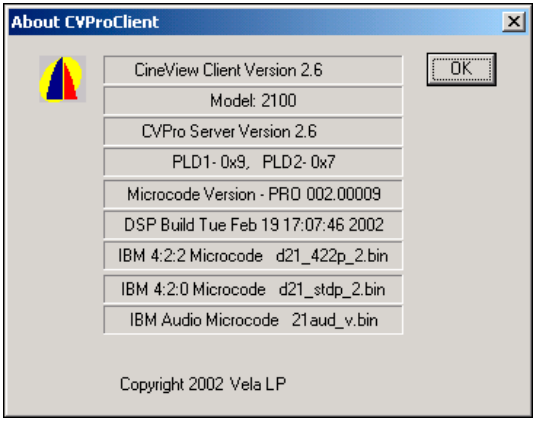


Figure 1-1. CineView Pro Help About Window

Other CineView Pro/Pro LE Decoder Features

- SMPTE 259M 525/625 component serial digital video output via standard BNC connector (CineView Pro XL and Pro only).
- Composite NTSC/PAL analog video output via standard BNC connector.
- Genlock input via standard BNC connector.
- Longitudinal Time Code is available at a header connector on the decoder board and can be output through an optional Vela add-on LTC connector bracket, or through a customer-fabricated cable assembly (Pro XL only).
- Built-in VGA video for host computer display.
- Selectable digital/analog stereo audio playback via a 15-pin high density connector (analog level is +4dBm = 0Vu). Digital audio format choices include AES/EBU at 110-ohm impedance or AES at 75-ohm impedance. (Digital audio output features are applicable to CineView Pro XL and CineView Pro only.)
- Other digital audio features include four channels of audio embedded in the SDI stream. Compressed S/P DIF audio is available at a header

connector on the decoder board and can be output if, desired, through a customer-fabricated cable assembly (CineView Pro XL only).

- MPEG audio layers 1 and 2 at 32kHz, 44.1kHz, and 48kHz.
- Mid-stream start: This feature gives the user the ability to begin playback of a video clip at any point. The keying function of mid-stream starts is user-definable and is determined by using byte offset. By using mid-stream start, the user can quickly verify the overall quality of a long encode. Mid-stream start operates by means of an easy-to-use slider bar within the CineView Pro and CineView Pro LE playback application.
- On-board hardware stream parsing.
- MPEG data rates up to 50Mbps (system dependant).
- Decoding of the following NTSC and PAL image resolutions:

NTSC	PAL
352×240 (SIF)	352×288 (SIF)
352×480 (Half D-1)	352×576 (Half Horizontal)
480×480	480×576
544×480	544×576
640×480	704×576
704×480	720×576 (PAL Full Resolution)
720×480 (CCIR-601 Resolution)	—

- Supported stream formats:
  - MPEG-1 elementary streams
  - MPEG-1 system streams
  - MPEG-2 elementary streams
  - MPEG-2 packetized elementary streams (PES)
  - MPEG-2 program streams
  - MPEG-2 transport streams
- Supports MPEG-1 video compression encapsulated in MPEG-2 program and transport streams.
- Reconstruction of I, P, and B frames.
- MPEG audio layers 1 and 2 at 32kHz, 44.1kHz, and 48kHz.

- Brightness, contrast, and saturation control for the VGA display.

Minimum System Requirements

- PC-compatible computer with Pentium 200 or better processor.
- 32MB RAM (64MB recommended).
- PCI bus architecture.
- Microsoft Windows 2000 or Windows NT 4.0 (Service Pack 6a).
- CD-ROM drive for software and driver installation.
- PCI-compliant VGA controller with accessible linear frame buffer (support for high bit 16 or high bit 24 required).
- Fast/Wide SCSI hard drive (recommended) or EIDE hard drive.

Hardware Installation

WARNING!

Hazardous electrical voltages may be present in your computer when its cover is removed for installation of this decoder. Follow proper safety procedures during installation. Remove all power from the computer before installing the decoder board.

STATIC DISCHARGE CAUTION

To avoid damage to the decoder from static discharge, keep the board in its protective bag until ready to be installed. Use an antistatic strap at all times when handling the decoder board during installation. Avoid touching components and edge connectors.

CineView Pro XL Hardware Installation

1. Power off the PC, unplug it from the AC source, and remove the cover to expose the chassis and motherboard.

It is important that the CineView Pro XL decoder board be plugged into a PCI bus master slot, making sure the edge connectors are fully engaged. Secure the board's mounting bracket to the PC chassis. Connect the audio and video cables, if applicable.

2. Connect the supplied audio output cable to the DB-15 connector on the decoder board. See Figure 1-28 for pinout information.

NOTE: The analog audio output level of the CineView Pro XL decoder is +4 dBm, balanced. Typical audio input levels for consumer television monitors are -10 dBm, unbalanced. This can lead to situations where the decoder's output level may be too high for the destination equipment, such as a common television monitor. For these instances, a cable with a T-type audio pad is provided. This will provide an unbalanced signal at a lower output level. For an illustration of the CineView Pro XL audio T-pad, refer to Figure 1-29.

3. If desired, S/P DIF audio can be acquired by attaching a connector to headers JP8 and JP9 on the CineView Pro XL board. Pinout information can be found in Table 1-1. Terminate the output in a connector of choice.

4. Connect video cables as desired. Refer to the CineView Pro XL board layout drawing at the end of this chapter.

5. LTC time code data can be acquired by attaching a connector to header JP9 on the decoder board. Pinout information can be found in Table 1-2. The LTC signal terminates in pins 1, 2 and 3 of the DB-9 connector on the PC bracket assembly (available from Vela). To simplify installation, this assembly includes the header connector and cabling.

6. Power up the PC as usual. Do not dismiss the "Update Device Driver" or "New Hardware Found" message boxes that appear. They will be helpful during the software installation, described in the sections that follow.

Header Connectors JP8/JP10, S/P DIF Output		
JP8	**Pin**	**Signal**
	1	S/P DIF, Chans. 1 & 2 Positive (+)
	2	Gnd
	3	S/P DIF, Chans. 1 & 2 Negative (–)
JP10	**Pin**	**Signal**
	1	S/P DIF, Chans. 3 & 4 Positive (+)
	2	Gnd
	3	S/P DIF, Chans. 3 & 4 Negative (–)

Table 1-1. S/P DIF Audio Output Header Pinouts

LTC Output Pinouts		
Header JP9 Pin	**Signal**	**DB-9 Connector Pin**
1	LTC Positive (+)	1
2	Gnd	2
3	LTC Negative (–)	3

Table 1-2. LTC Output Header Pinouts

Digital Audio Jumper Settings

The CineView Pro XL decoder board provides two jumper connector blocks that are used to configure the impedance of the AES/EBU digital audio signal. JP6 controls AES/EBU channel A. Jumper block JP7 controls AES/EBU channel B. Position the jumpers as shown in Figure 1-2 to set the output impedance of each of the channels. As noted in the drawing, both jumper blocks should be set identically, unless different output impedances are explicitly desired. AES/EBU audio terminates in a 15-pin high-density D-sub connector on the rear of the decoder board. Pinouts, signal information, and a view of the connector can be found in Figure 1-28 at the end of this chapter.

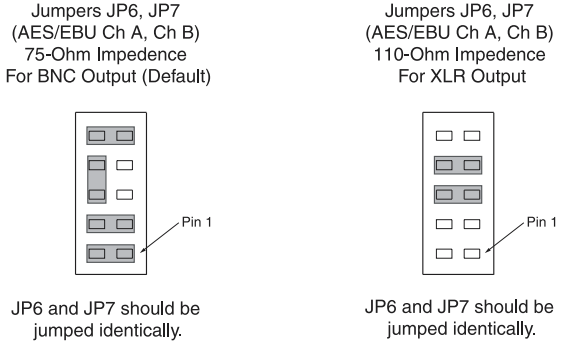


Figure 1-2. AES/EBU Jumpers

CineView Pro/Pro LE Hardware Installation

1. Power off the PC, unplug it from the AC source, and remove the cover to

expose the chassis and motherboard.

2. It is important that the CineView Pro/Pro LE decoder board be plugged into a PCI bus-master slot, making sure the edge connectors are fully engaged. Secure the board's mounting bracket to the PC chassis. Connect the audio and video cables, if applicable.

3. Connect the supplied audio output cable to the DB-15 connector on the decoder board, through which two channels of balanced analog audio at +4dBm and/or one channel of digital (AES) audio can be obtained. (Note that the CineView Pro LE decoder does not support digital audio output.) Outputs are terminated in male XLR connectors. See Figure 1-27 and Figure 1-28 for audio pinouts.

NOTE: The audio output level of the CineView Pro/Pro LE decoder is +4 dBm, balanced. Typical audio input levels for consumer television monitors are -10 dBm, unbalanced. This can lead to situations where the decoder's output level may be too high for the destination equipment, such as a common television monitor. For these instances, a cable with a T-type audio pad is provided. This will provide an unbalanced signal at a lower output level. For an illustration of the CineView Pro/Pro LE audio T-pad, refer to Figure 1-29.

4. Connect video cables as desired. Refer to the CineView Pro and CineView Pro LE decoder board layout drawings at the end of this chapter.

5. Power up the PC as usual. Do not dismiss the "Update Device Driver" or "New Hardware Found" message boxes that may appear. They will be helpful during the software installation, described in the sections that follow.

Software Installation

The following sections pertain to the installation of Vela Release 2.6 system software. Determine the operating system you are using (Windows 2000 or Windows NT) and follow the corresponding instructions.

Be sure to review the "readme" files on the system software CD-ROM for the very latest information on installation and performance issues.

Installing Under Windows NT

All of the software that you need to install the CineView Pro series decoder on Windows NT 4.0 is located on the single CD-ROM you received with the decoder. Just follow these steps, whether you are upgrading to version 2.6 or installing it for the first time. Remember that a number of system restarts (reboots) may be required during the installation process. **Do not remove the CD-ROM disc from the drive until you've completed the last step of the installation process.**

NOTE: Hardware drivers for Windows NT are automatically installed with the application.

1. Uninstall any CineView Pro software currently on the system. Use the Windows Control Panel > Add/Remove Programs application. For complete details on uninstalling current and previous versions of software, see "Uninstalling CineView Pro 2.6 Software," page 24.

2. Install Windows NT Service Pack 6a or later, if it's not already installed on your system. You can download this service pack from the Internet, or you can install it from the CineView Pro System Software CD-ROM, as follows:

- Insert the CD-ROM into the CD-ROM Drive. The Autorun install setup screen will appear (Figure 1-10).
- From the Autorun screen, select the "Explore This CD" option. A screen similar to that in Figure 1-11 will appear.
- Double-click on the System folder.
- Double-click on the "NT Service Pack" folder.
- Double-click on the **sp6i386.exe** file.
- The application will extract files, then ask you to read a license agreement. After reading the agreement, check the "Accept License Agreement" checkbox as well as the "Backup Files Required to Uninstall" checkbox.
- Click Install.
- Click Next, OK, or Finished to all of the screens that follow.
- If you are asked to reboot, allow the system to restart before continuing.

3. The Autorun setup screen (Figure 1-10) should appear after the reboot. If it does not, use Windows Explorer to select and run **setup.exe** from the CD-ROM.

4. Install Internet Explorer version 5, Service Pack 4 or later, if it's not already installed on your system. Again, you can download this from the Internet, or you can install it from the Vela system software installation CD-ROM. To use the CD-ROM, follow these steps:

- Insert the CD-ROM disc into the CD-ROM drive.
- When the setup screen appears, select the "Explore This CD" option.
- Double-click on the "System" folder.
- Double-click on the "MSIE5 NT" folder.
- Double-click on the setup.exe icon. When a window pops up to select "Typical" or "Minimal" Installation, select "Minimal."

- On the next screen, deselect (remove the checks from) the “Windows Media Player” and “Outlook Express” selections.
- The version of Windows Media Player included with Internet Explorer 5.0 is outdated. A more recent version is included on this CD, though the installation of Windows Media Player and Outlook Express are **not** a requirement for the CineView Pro.
- Click Next, OK, or Finished to all of the screens that follow.
- If you are asked to reboot, allow the system to restart before continuing.

5. The setup screen should appear after the reboot. If it does not, use Windows Explorer to select and run **setup.exe** from the CD-ROM or click Start > Run.

6. Install MDAC 2.6, if it is not already installed on your system. Again you can download this from the Internet, or you can install it from the installation CD-ROM. To use the CD-ROM, follow these steps:

- Insert the CD-ROM into the CD-ROM drive.
- When the setup screen appears, select the “Explore This CD” option.
- Double-click on the “System” folder.
- Double-click on the “MDAC 2.6" folder.
- Double-click on the **mdac_typ.exe**.
- The application will extract files, then ask you to read a license agreement. After reading the agreement, check the “Accept License Agreement” checkbox. Click Next.
- If you are asked to restart, allow the system to reboot before continuing.

7. The Autorun setup screen (Figure 1-10) should appear after the system restart. If it does not, use Windows Explorer to select and run **setup.exe** from the CD-ROM or click Start > Run.

8. From the Autorun setup screen, select Install “CineView Pro/LE/XL or CineView Pro Prism,” then follow the steps listed below:

- Read the Welcome screen (Figure 1-12), then click Next.
- On the “Choose Destination Location” screen (Figure 1-13), accept the C:\Program Files\ Vela Research destination, as listed, by clicking Next. **Do not change the destination!** The default destination is required for proper decoder operation.

9. On the “Select Components” screen (Figure 1-14):

- If you have not yet run the MFC update option for this release, check the

"MFC Update" checkbox on the "Select Components" screen. This step needs to be done only once. If you check this box, there will be an extra set of steps automatically included in the procedure specified below— and the installation procedure will require an additional system restart. Just follow the instructions presented by the MFCupdate.exe installation application.

- Also, under Required Components on the "Select Components" screen, check the "Core Decoder Modules" checkbox.
- If you have purchased any of the following components, their corresponding check boxes also need to be checked.

  Optional Components:

  JTAG Update Utility
  Direct Show Filter (Beta)

  Optional CineView Pro Prism Components:

  Ligos Module (MPEG-1)
  Real Networks Module (G2)
  Microsoft Module (WMF)

  SDK:

  CineView Pro SDK
  CineView Pro SDK — VB Sample Application
  CineView Pro SDK — Non-Unicode DLL
  CineView Pro Prism SDK

  Because they are password-protected, you will be able to install these applications only if you purchased one or all of them and received the corresponding password. If you cannot locate the password, contact Vela support for assistance. You can return to this install screen at a later time if needed.)
- Click Next to proceed with the installation of the selected components.
- On the "Select Program Manager Group" screen (Figure 1-15), accept Vela Research by clicking Next.
- On the "Start Installation" screen (Figure 1-16), click Next.
- A "DO NOT REMOVE THE CD" message box will display as a reminder that a number of reboots may be required during the installation process. Click OK to continue.
- If you have chosen to run the MFC Update option, the installation process will begin here to copy files.

- On the “Install” message box (Figure 1-17), note that the system must be restarted. Click OK, and then wait as the system reboots. **Leave the CD-ROM in the drive through the system restart process.**

10. If you remembered to leave the CD-ROM in the drive, the setup application pops up immediately after the reboot. Continue with the installation by following these steps:

- On the “CineView Pro End User License Agreement” screen (Figure 1-19), select the “I Agree” radio button after reading the agreement. Then click OK. The application will install some files.

11. If you elected to install the SDK:

- Select the “I Agree” radio button on the “CineView Pro SDK End User License Agreement” screen (Figure 1-20). Click OK.
- On the Password screen (Figure 1-21), you will be asked for a password. Use the one supplied when you purchased your Software Developer’s Kit. If you have problems finding your password, contact Vela Support. After entering the password, click OK. A few more files will be installed.

12. On the “Installation Complete” screen (Figure 1-22), note that CineView Pro 2.6 has been successfully installed. Click Finish.

13. The “Install” message box (Figure 1-17) will note that the system must be re-started. Click OK, then let the system reboot. **Leave the CD-ROM in the drive**.

14. After the system has been rebooted, close the setup application if it is active, then remove the CD-ROM from the drive.

Installing Under Windows 2000

Just follow these steps to upgrade to version 2.6, remembering that a number of reboots may be required during the installation process. D**o not remove the CD-ROM disc from the drive until you've completed the final step.**

Note: Do not install any new hardware at this time.

The following instructions are intended for a clean Windows 2000 PC with no CineView Pro currently installed.

If CineView Pro decoder family system software is currently installed, first uninstall the software from the Control Panel. See “Uninstalling CineView Pro 2.6 Software,” page 24.

1. Install Windows 2000 Service Pack 2, if it is not already installed on your system. You can download this service pack from the Internet, or you can install it from the CD-ROM. To use the CD-ROM, follow these steps:

- From the Autorun screen (Figure 1-10), select “Explore This CD.”
- From the “Explore This CD” screen (Figure 1-11), double-click on the System folder.
- Double-click on the “Win2k Service Pack” folder.
- Double-click on the **w2ksp2.exe** file.
- The application will extract files, then ask you to read a license agreement. After reading the agreement, check the “Accept License Agreement” check-box as well as the “Backup Files Required to Uninstall” checkbox.
- Click Install.
- Click Next, OK, or Finished to all of the screens that follow.
- If you are asked to reboot, allow the system to restart before continuing.

2. The Autorun screen should appear after the reboot. If it does not, run **setup.exe** from the CD-ROM or click Start > Run.

After installing Windows 2000 Service Pack 2, if necessary, follow the instructions below to install hardware drivers. after installing any Vela hardware. Remember that a number of reboots may be required during the install process.

Hardware Driver Installation — Windows 2000

Before installing CineView Pro Release 2.6 system software, and after installing the Windows 2000 Service Pack 2, follow these directions to install the hardware drivers:

1. Shut down Windows, power off the system, and install the new hardware.
2. After reassembling the machine, power up the system as usual.
3. If the “Add New Hardware Wizard” appears, click Cancel.
4. Insert the CD-ROM if you have not already done so. Exit the Autorun setup screen if it appears.
5. Right click on “My Computer.” A drop-down menu should appear.
6. Highlight “Properties,” then click on it.
7. Click on the tab labeled “Hardware.”
8. Click on “Device Manager.” A screen similar to that of Figure 1-3 appears.
9. Right click on the following option:
   - Multimedia Controller (for the CineView Pro family decoders).
10. Select Properties from the small drop-down menu that appears.

11. Click on “Reinstall Driver.”

12. The Device Driver Wizard screen appears (Figure 1-4). “Welcome to the Upgrade Device Driver Wizard.” Click Next.

13. Wizard screen “Install Hardware Device Drivers” (Figure 1-5): Select the radio button “Search for a suitable driver for my device.” Click Next.

14. Wizard screen “Locate Driver Files” (Figure 1-6): Check “Specify a Location,” uncheck everything else and Click Next, or if your installing the software from the CD, select the CD-ROM drives option. If you choose this second option you can skip the next step.

15. A message box (Figure 1-7) will appear that will allow you to browse your computer until you find a driver for the Multimedia controller. Check Browse to select the device from the following path for the driver.

- The hardware driver for the Multimedia Controller will be found at \drivers\cineviewpro\windows 2000\saa7146.inf

16. Click OK.

17. Repeat steps 9 – 14 for the other Unknown Multimedia device in step 9. The steps must be repeated for each CineView Pro hardware device.

18. When the Vela Encoder driver installation is complete, a prompt screen (Figure 1-8) will appear: “Completing the Upgrade Device Driver Wizard.” Click Finish.

19. At this time you will receive a “System Settings Change” message box (Figure 1-9). Click Yes to restart the system. If this does not occur, manually reboot your system. **Do not remove the CD-ROM disc during the restart process.**

20. The setup screen should pop up after the system restart. If it does not, run **setup.exe** from the CD-ROM or click Start > Run.

Proceed with the installation of CineView Pro system software. See “Installing Under Windows 2000” on page 15.

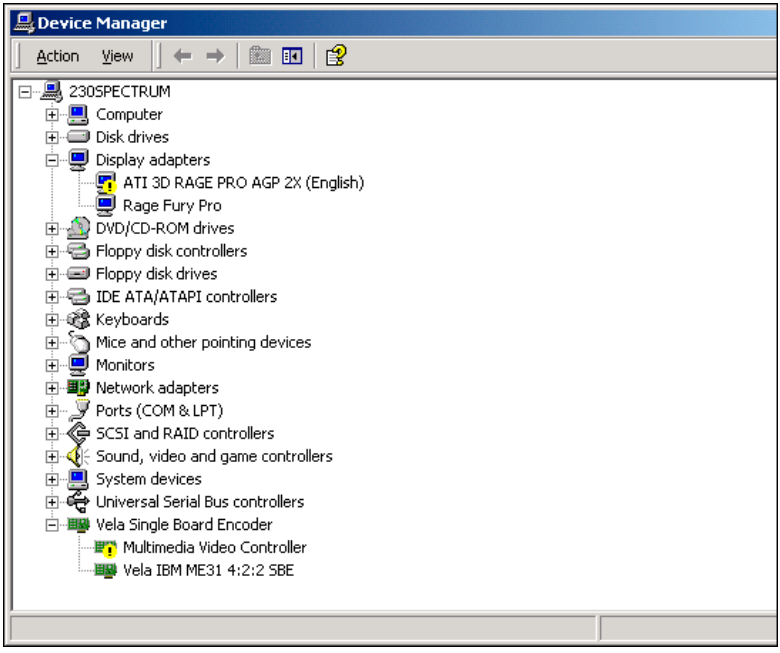


Figure 1-3. System Properties Device Manager Screen

Figure 1-4. Device Driver Wizard: Welcome Screen

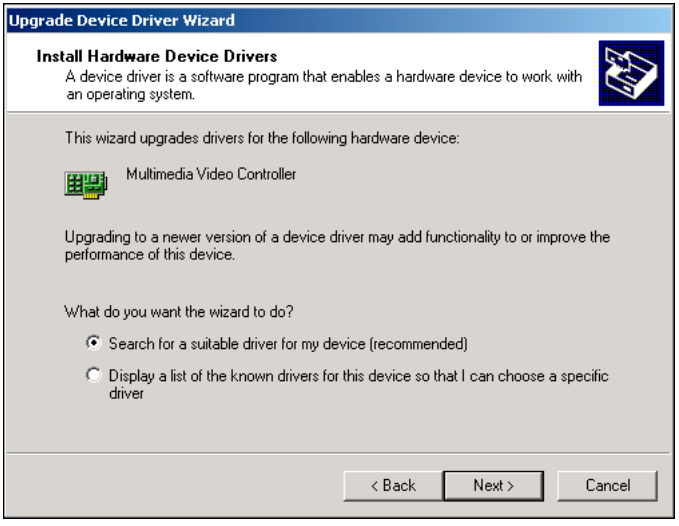


Figure 1-5. Device Driver Wizard: Install Drivers

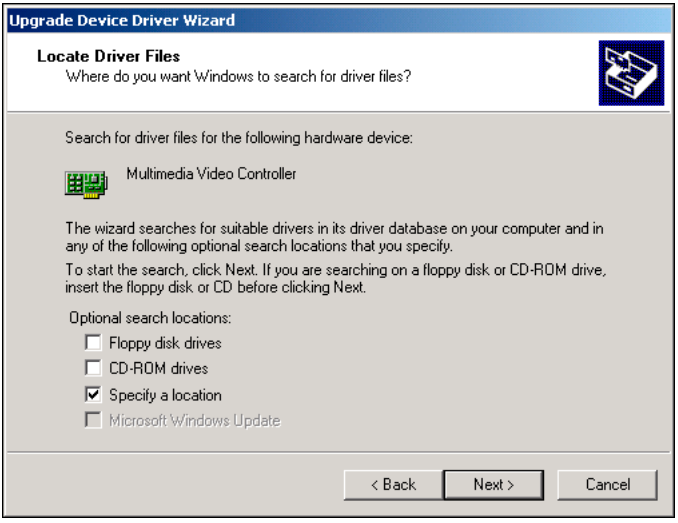


Figure 1-6. Device Driver Wizard: Locate Driver Files

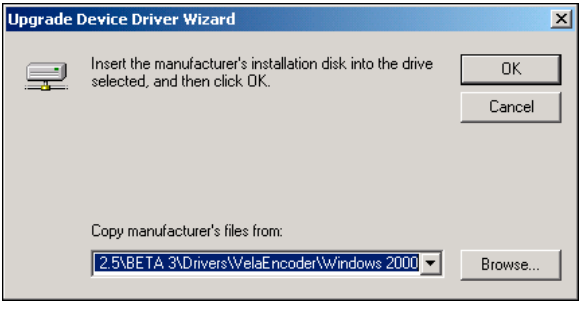


Figure 1-7. Device Driver Wizard: Browse Window

Figure 1-8. Device Driver Wizard: Completion

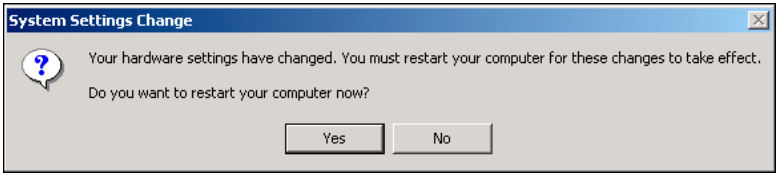


Figure 1-9. System Settings Change Message Box

Installing CineView Pro 2.6 Software — Windows 2000

1. Install MDAC 2.6, if it is not already installed on your system. Again you can download this from the Internet, or you can install it from the installation CD-ROM. To use the CD-ROM, follow these steps:

- Insert the CD-ROM into the CD-ROM drive.
- When the Autorun setup screen (Figure 1-10) appears, select the “Explore This CD” option (Figure 1-11).
- Double-click on the “System” folder.
- Double-click on the “MDAC 2.6" folder.
- Double-click on the **mdac_typ.exe**.
- The application will extract files, then ask you to read a license agreement. After reading the agreement, check the “Accept License Agreement” checkbox. Click Next.
- If you are asked to restart, allow the system to reboot before continuing.

2. The Autorun setup screen (Figure 1-10) should appear after the system restart. If it does not, use Windows Explorer to select and run **setup.exe** from the CD-ROM or click Start > Run.

3. From the Autorun setup screen, select Install “CineView Pro/LE/XL or CineView Pro Prism,” then follow the steps listed below:

- Read the Welcome screen (Figure 1-12), then click Next.
- On the “Choose Destination Location” screen (Figure 1-13), accept the C:\Program Files\ Vela Research destination, as listed, by clicking Next. **Do not change the destination!** The default destination is required for proper decoder operation.

4. On the “Select Components” screen (Figure 1-14):

- If you have not yet run the MFC update option for this release, check the “MFC Update” checkbox on the “Select Components” screen. This step needs to be done only once. If you check this box, there will be an extra set of steps automatically included in the procedure specified below— and the installation procedure will require an additional system restart. Just follow the instructions presented by the MFCupdate.exe installation application.
- Under Required Components on the “Select Components” screen, check the “Core Decoder Modules” checkbox.

- If you have purchased any of the following components, their corresponding check boxes also need to be checked.

  Optional Components:

  JTAG Update Utility
  Direct Show Filter (Beta)

  Optional CineView Pro Prism Components:

  Ligos Module (MPEG-1)
  Real Networks Module (G2)
  Microsoft Module (WMF)

  SDK: CineView Pro SDK
  CineView Pro SDK — VB Sample Application
  CineView Pro SDK — Non-Unicode DLL
  CineView Pro Prism SDK

  Because they are password-protected, you will be able to install these applications only if you purchased one or all of them and received the corresponding password. If you cannot locate the password, contact Vela support for assistance. You can return to this install screen at a later time if needed.

- Click Next to proceed with the installation of the selected components.
- On the "Select Program Manager Group" screen (Figure 1-15), accept Vela Research by clicking Next.
- On the "Start Installation" screen (Figure 1-16), click Next.
- A "DO NOT REMOVE THE CD" message box will display as a reminder that a number of reboots may be required during the installation process. Click OK to continue.
- If you have chosen to run the MFC Update option, the installation process will begin here to copy files. Note: If you have not already done so, you **must** run the "MFC Update" and "Core Decoder Modules" under Required Components. These two check-boxes must always be checked when requesting the SDK — **This insures proper installation of the SDK**.
- On the "Install" message box (Figure 1-17), note that the system must be restarted. Click OK, and then wait as the system reboots. **Leave the CD-ROM in the drive through the system restart process.**

5. If you remembered to leave the CD-ROM in the drive, the setup application pops up immediately after the reboot. Continue with the installation by following these steps:

- On the “CineView Pro End User License Agreement” screen (Figure 1-19), select the “I Agree” radio button after reading the agreement. Then click OK. The application will install some files.

6. If you elected to install the SDK:
   - Select the “I Agree” radio button on the “CineView Pro SDK End User License Agreement” screen (Figure 1-20). Click OK.
   - On the Password screen (Figure 1-21), you will be asked for a password. Use the one supplied when you purchased your Software Developer’s Kit. If you have problems finding your password, contact Vela Support. After entering the password, click OK. A few more files will be installed.

7. On the “Installation Complete” screen (Figure 1-22), note that CineView Pro 2.6 has been successfully installed. Click Finish.

8. The “Install” message box (Figure 1-17) will note that the system must be re-started. Click OK, then let the system reboot. **Leave the CD-ROM in the drive**.

9. After the system has been rebooted, close the setup application if it is active, then remove the CD-ROM from the drive.

Uninstalling CineView Pro 2.6 Software

If, at some point, you need to uninstall CineView Pro decoder family system software, always use the Windows Control Panel > Add/Remove Programs application. Never manually remove any files installed by the installation process, unless specifically directed, as doing so may prevent proper removal of some or all of the files installed. Note that complete removal of the product with Add/Remove Programs is possible only if the product was originally installed in the default target folder created by the install process. If you have CineView Pro software installed as part of an Argus encoding system, please refer to the *Argus Installation and User Manual* for instructions on uninstalling system software.

In most cases, when you upgrade to version 2.6, you'll be removing a previous version of CineView Pro software. To uninstall that previous version of software, consult the user's manual that you received when you installed it. At some point, you may need to uninstall CineView Pro version 2.6 from your system. To do so, just follow the steps below.

- On the software list that is displayed when you select “Add/Remove Programs” from the Control Panel, click on the entry that begins with the phrase “Vela Research, Software Version ….” Then, on the Uninstall Information panel, click OK.
- When asked to select the uninstall method, select “Automatic” and click Next.

- You may see a message indicating that a particular file is no longer being used by any program. Click “Yes to All” to delete it and any other such files.
- On other screens that may appear, Click on Next or Finish until the uninstall process is complete.
- After you click Finish on the last screen of the uninstall procedure, you will be asked to restart the system. Click OK, and allow the system to restart.

When the system has finished rebooting, there should be no file entries in the C:\Program Files\Vela Research folder. If you find files or folders remaining in the C:\Program Files\Vela Research folder, delete them manually.

Figure 1-10. Installation Autorun Setup Screen

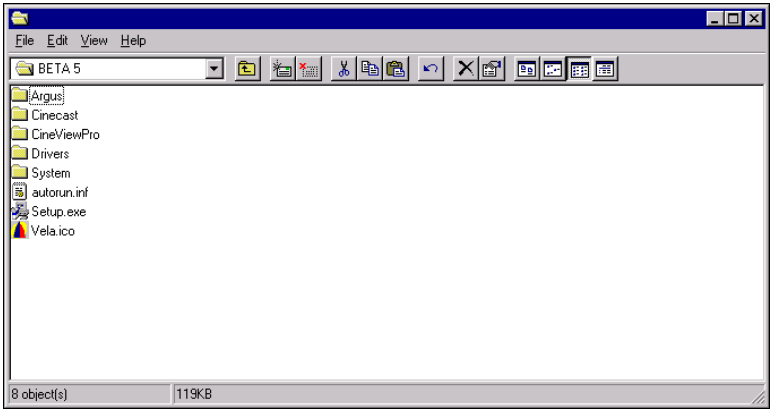


Figure 1-11. "Explore This CD" Screen

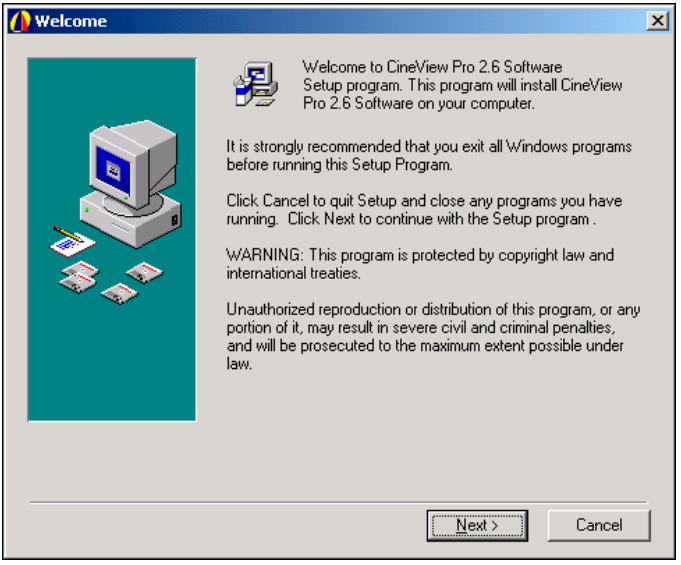


Figure 1-12.Install Welcome Screen

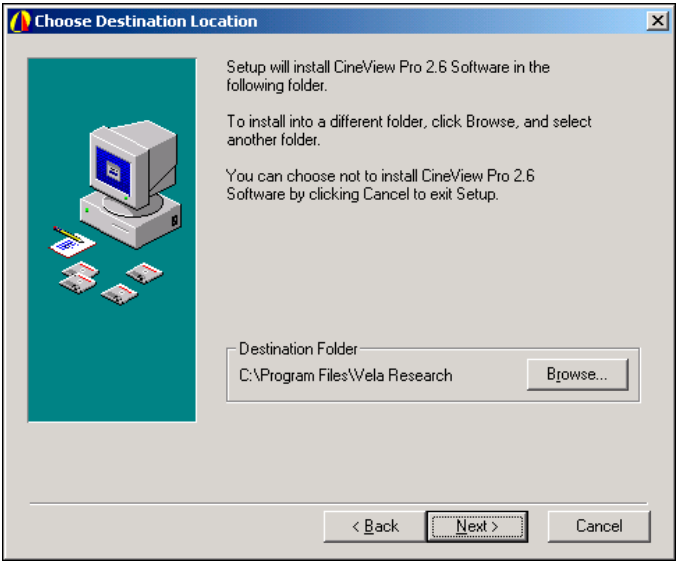


Figure 1-13.Destination Location Screen

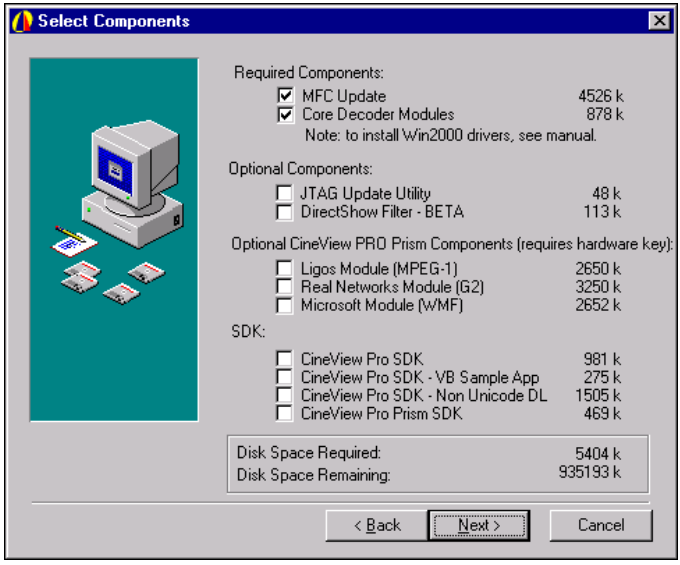


Figure 1-14.Select Components Screen

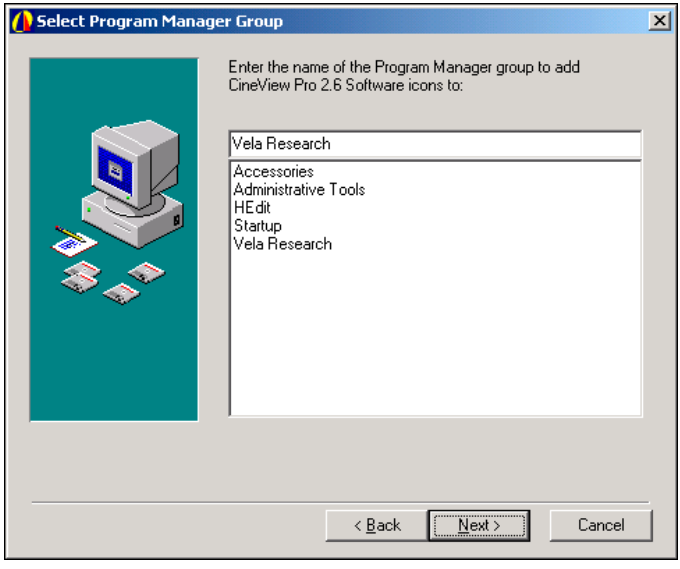


Figure 1-15.Program Manager Group Screen

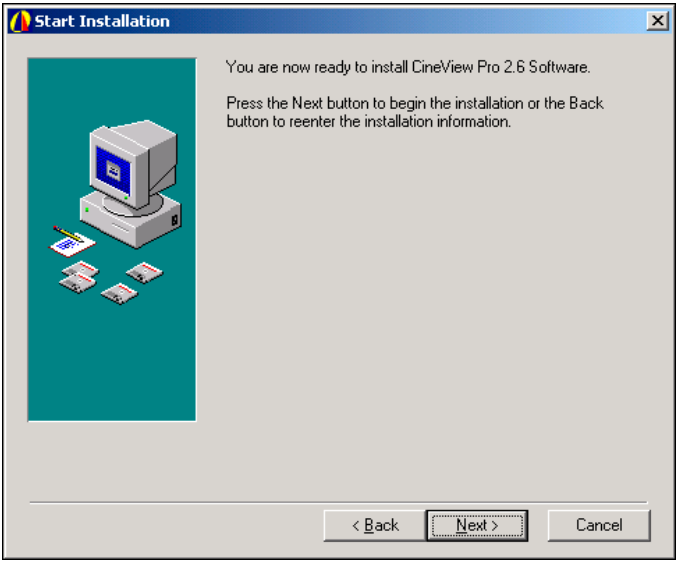


Figure 1-16.Installation Start Screen

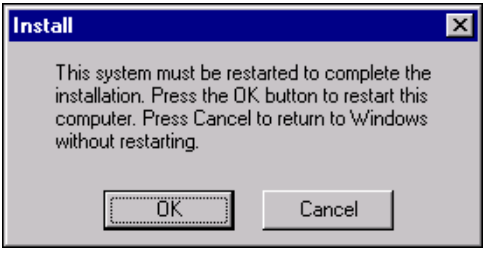


Figure 1-17.Install Restart Message

Figure 1-18.Registry Editor Information Screen

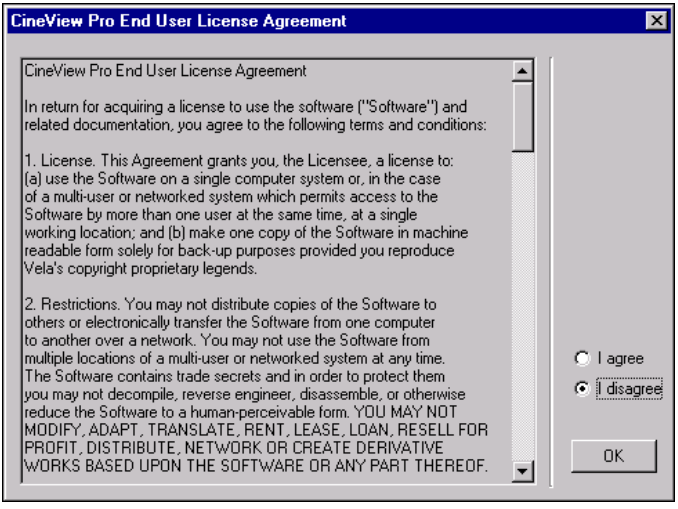


Figure 1-19.CineView License Agreement Screen

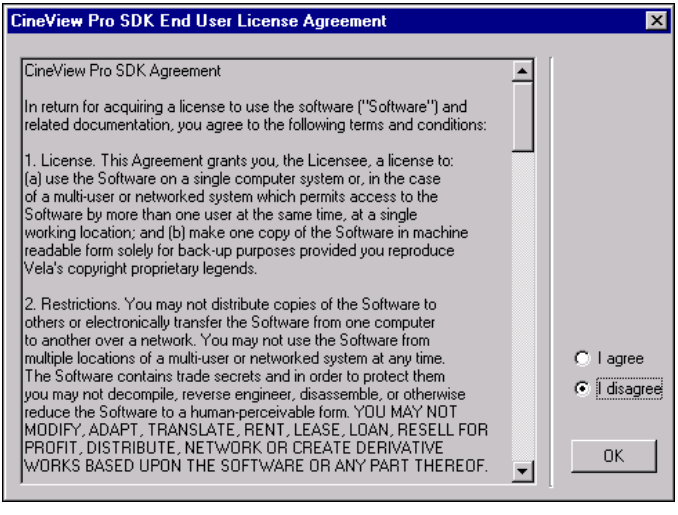


Figure 1-20.CineView SDK License Agreement Screen

Figure 1-21.Password Dialog Box

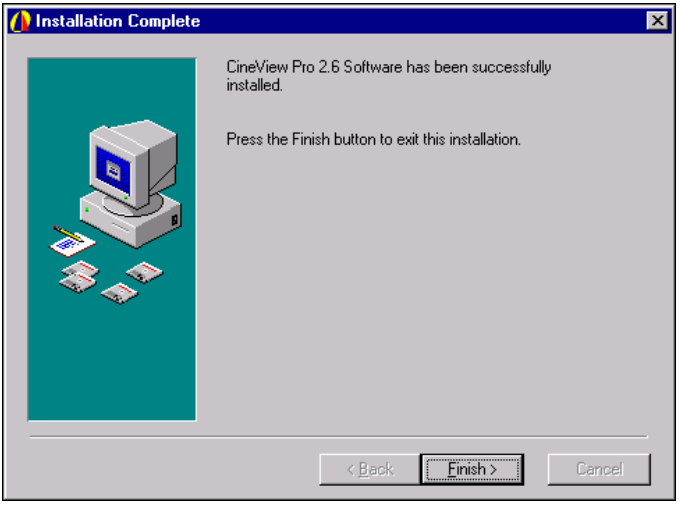


Figure 1-22.Installation Completion Screen

Windows Registry

For each CineView decoder card installed on the computer, the unique settings for each card are stored in the Windows Registry under HKEY_CURRENT_USER, Software, Vela Research, CineView Pro X, Setup; where X is the logical zero-based index of the CineView decoder card installed on the computer. See Figure 1-23, below.

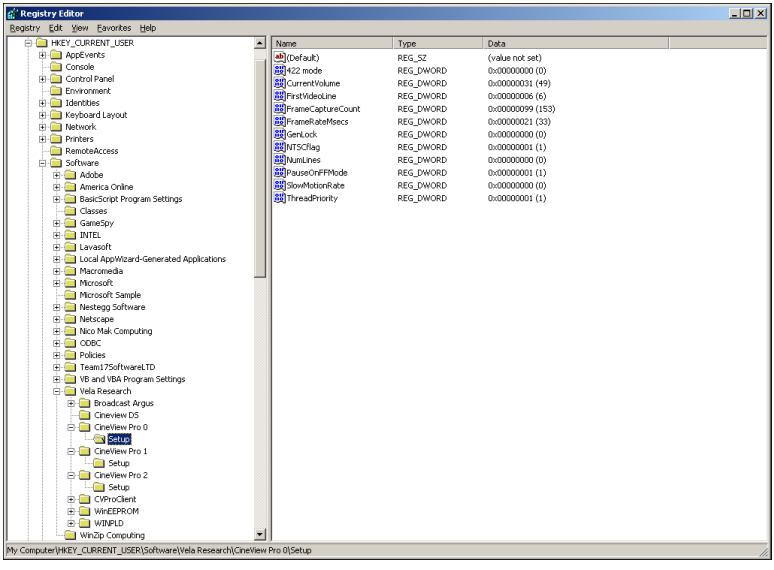


Figure 1-23. Microcode Path Listing in Registry Window

Customer Support

In the event of problems with your CineView Pro/Pro LE decoder, please contact the Vela Training and Support staff as follows:

- Phone: (727) 507-5301
- E-mail: support@vela.com
- World Wide Web — http://www.vela.com

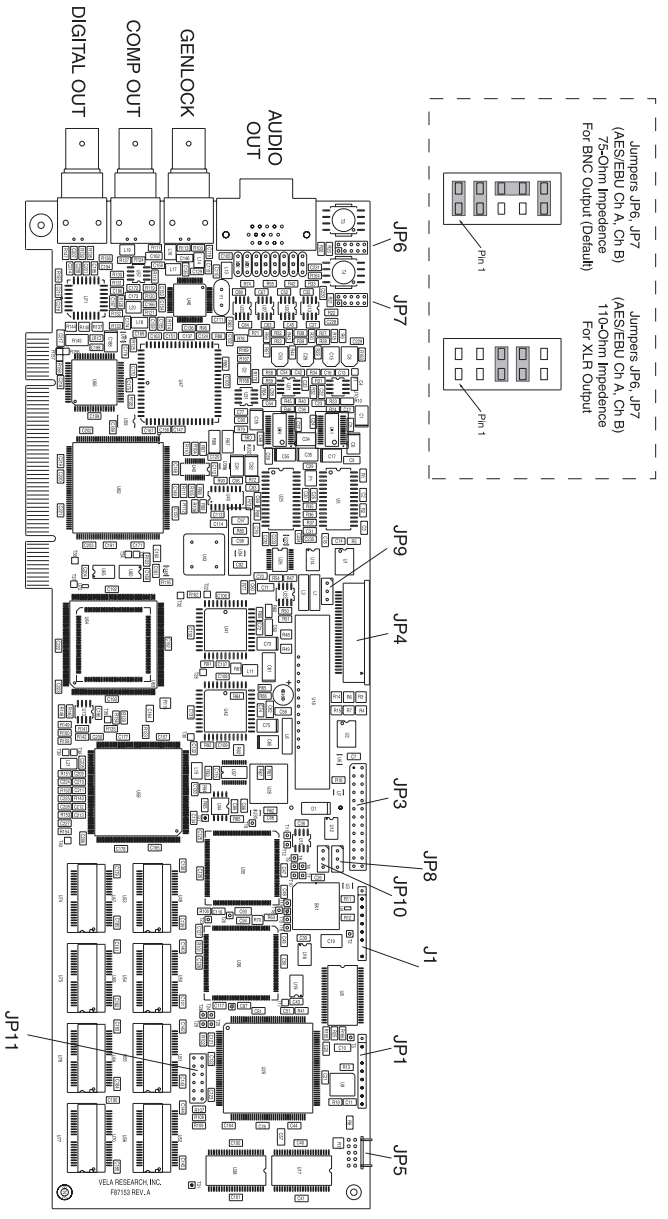


Figure 1-24. CineView Pro XL Decoder Board Layout

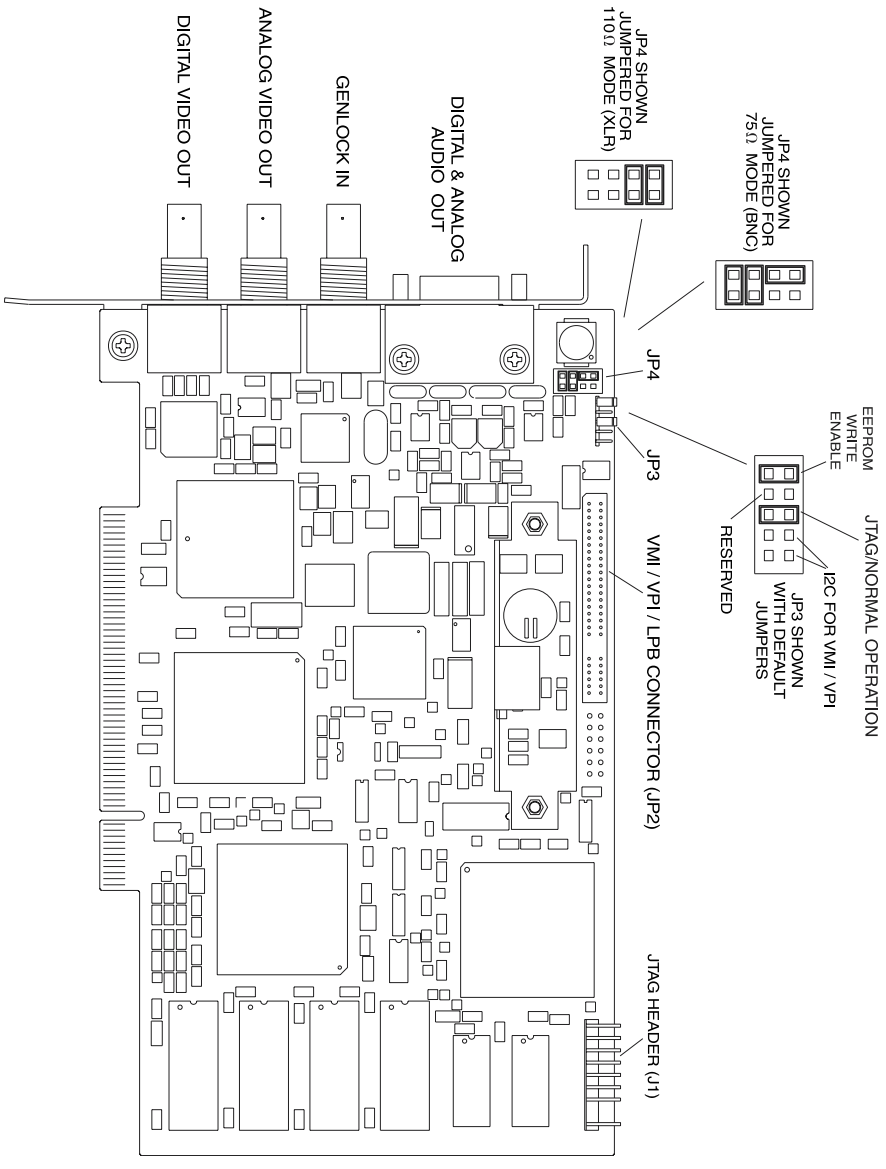


Figure 1-25. CineView Pro Decoder Board Layout

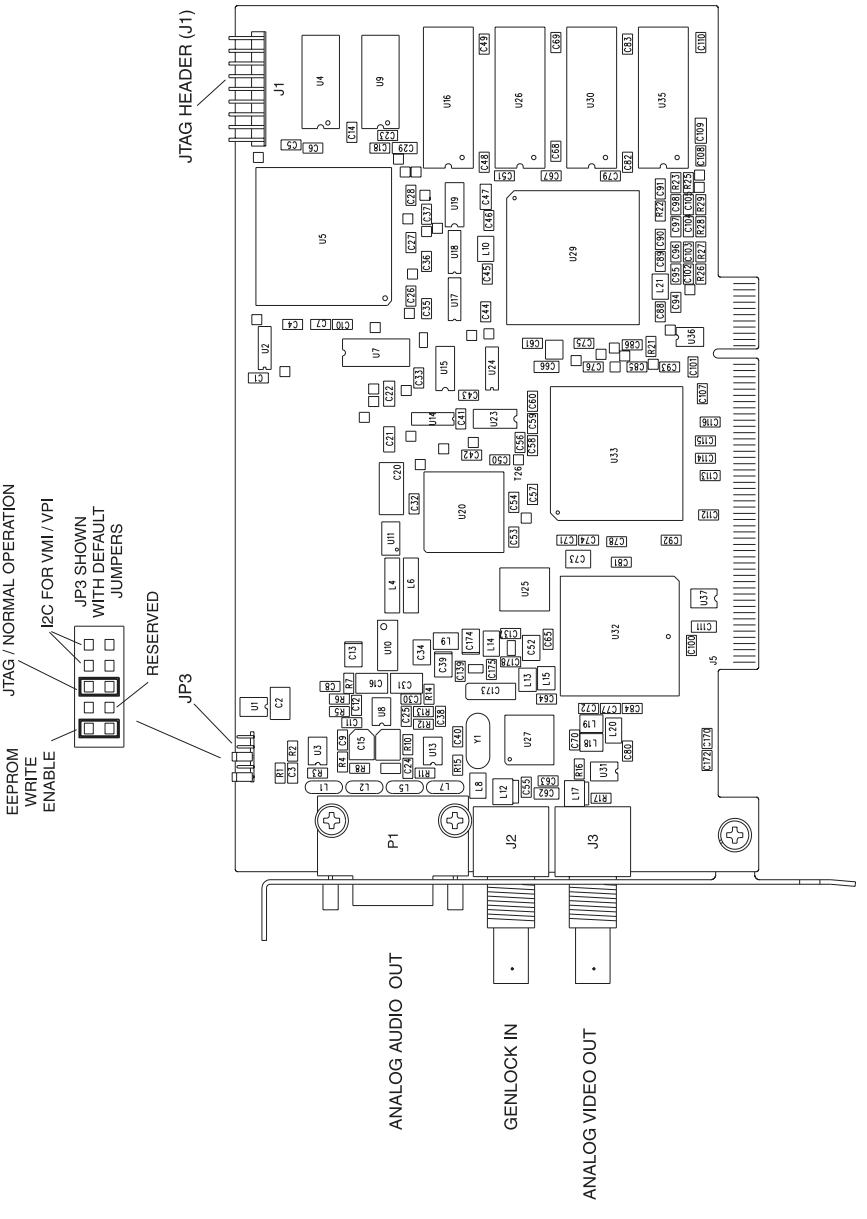
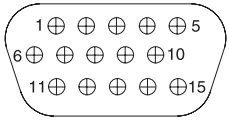


Figure 1-26. CineView Pro LE Decoder Board Layout

AUDIO CONNECTOR
15-PIN HIGH-DENSITY
SUB-D MATING CONNECTOR
(AMP P/N 748364-1 or EQUIV.)



FRONT (PIN) VIEW OF PLUG

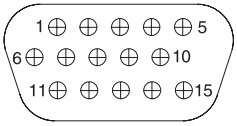
This connector mates with the 15-pin audio connector on the CineView Pro XL.

PINOUT DESCRIPTION

Pin 1 –	Analog Ch B (Left +)	(XLR Pin 2)
Pin 2 –	Analog Ch A (Left –)	(XLR Pin 3)
Pin 3 –	Gnd	(XLR Pin 1)
Pin 4 –	Analog Ch B (Right +)	(XLR Pin 2)
Pin 5 –	Analog Ch A (Right +)	(XLR Pin 2)
Pin 6 –	Analog Ch B (Left –)	(XLR Pin 3)
Pin 7 –	Analog Ch A (Left +)	(XLR Pin 2)
Pin 8 –	Gnd	(XLR Pin 1)
Pin 9 –	Analog Ch B (Right –)	(XLR Pin 3)
Pin 10 –	Analog Ch A (Right –)	(XLR Pin 3)
Pin 11 –	Gnd	(XLR Pin 1)
Pin 12 –	AES Ch B +	(XLR Pin 2)
Pin 13 –	AES Ch B –	(XLR Pin 3)
Pin 14 –	AES Ch A –	(XLR Pin 3)
Pin 15 –	AES Ch A +	(XLR Pin 2)

Figure 1-27. CineView Pro XL Audio Pinouts

AUDIO CONNECTOR
15-PIN HIGH-DENSITY
SUB-D MATING CONNECTOR
(AMP P/N 748364-1 or EQUIV.)



FRONT (PIN) VIEW OF PLUG

This connector mates with the audio connector on the CineView Pro / Pro LE decoder board.

PINOUT DESCRIPTION

Pin 1 – Reserved
Pin 2 – Analog chan. 1 (left –)
Pin 3 – Gnd
Pin 4 – Reserved
Pin 5 – Analog chan. 2 (right +)
Pin 6 – Reserved
Pin 7 – Analog chan. 1 (left +)
Pin 8 – Gnd
Pin 9 – Reserved
Pin 10 – Analog chan. 2 (right –)
Pin 11 – Reserved
Pin 12 – Reserved
Pin 13 – Reserved
Pin 14 – AES Chans. 1, 2 (–)
Pin 15 – AES Chans. 1, 2 (+)

Pins 14 & 15 Applicable to CineView Pro Only

Figure 1-28. CineView Pro/Pro LE Audio Pinouts

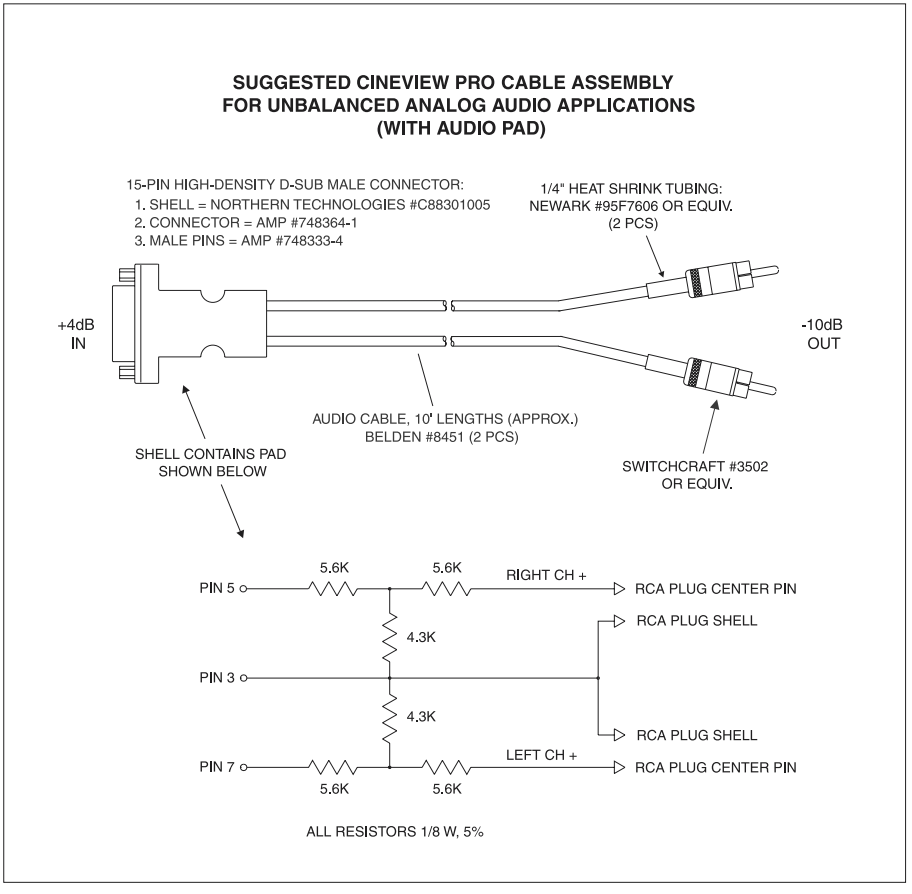


Figure 1-29. Suggested Audio Cabling (Unbalanced Analog)

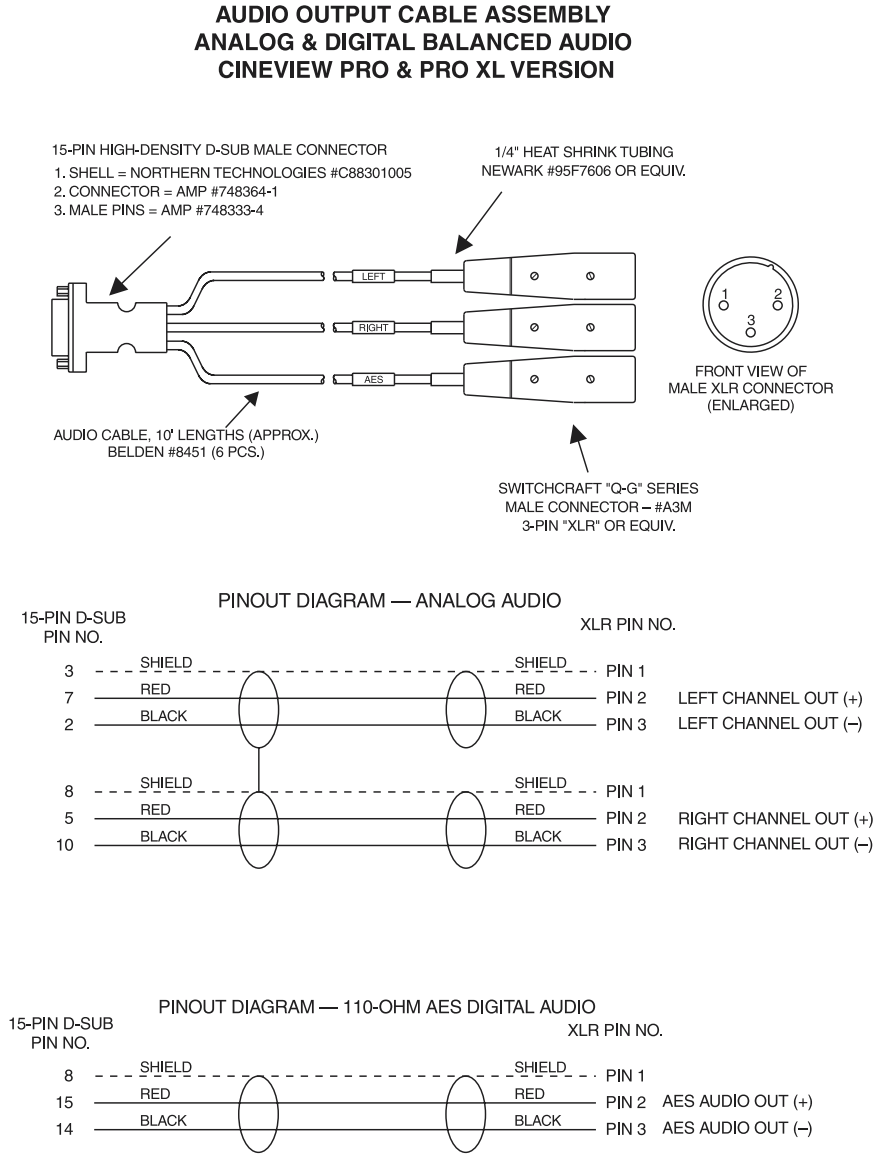


Figure 1-30. Suggested Audio Cabling (Balanced Analog), CineView Pro & XL

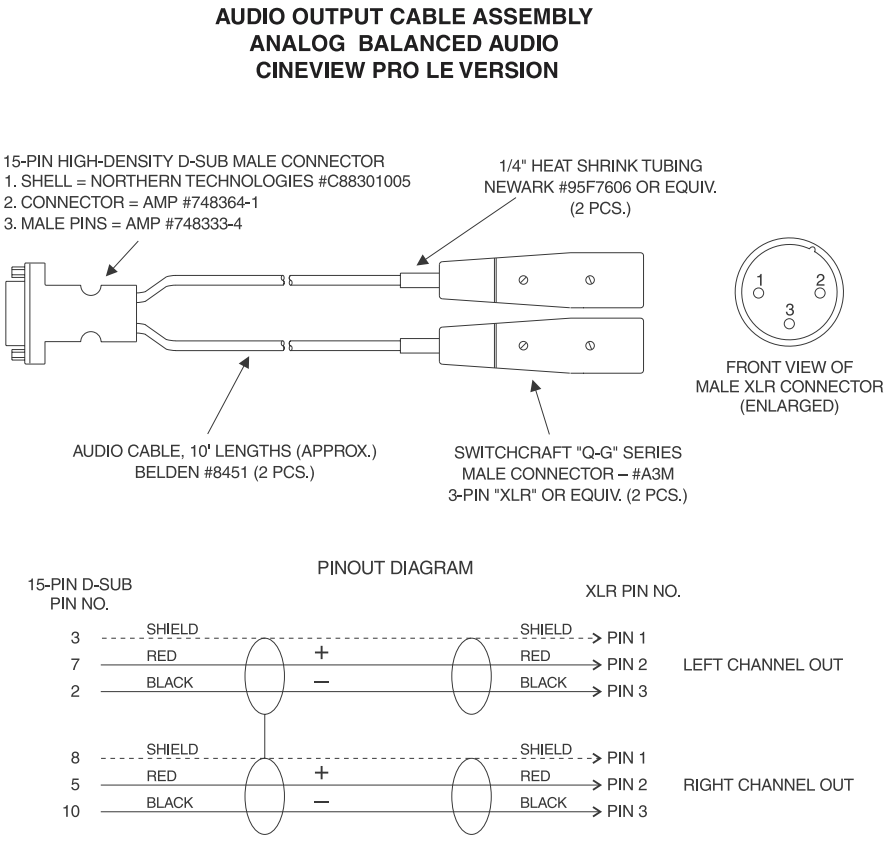


Figure 1-31. Suggested Audio Cabling (Balanced Analog), CineView Pro LE

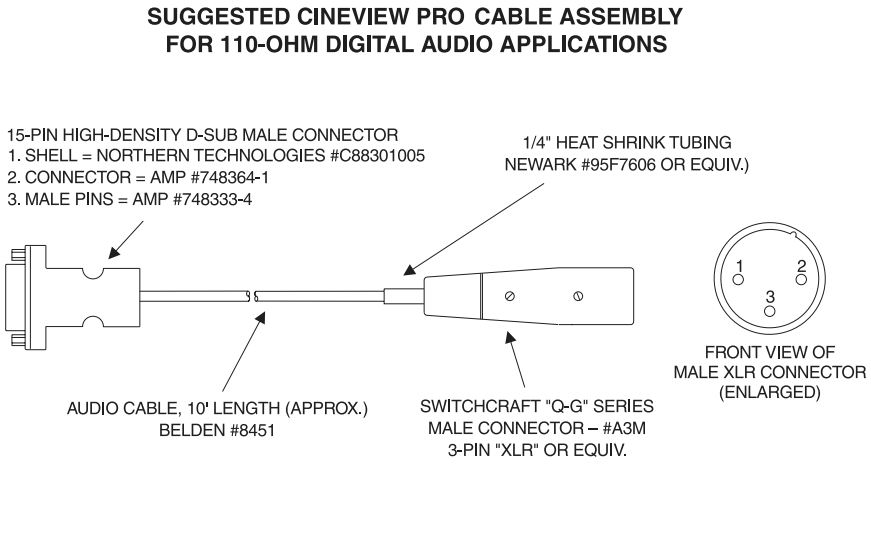


Figure 1-32. Suggested Audio Cabling (Digital 110-Ohm), CineView Pro & XL

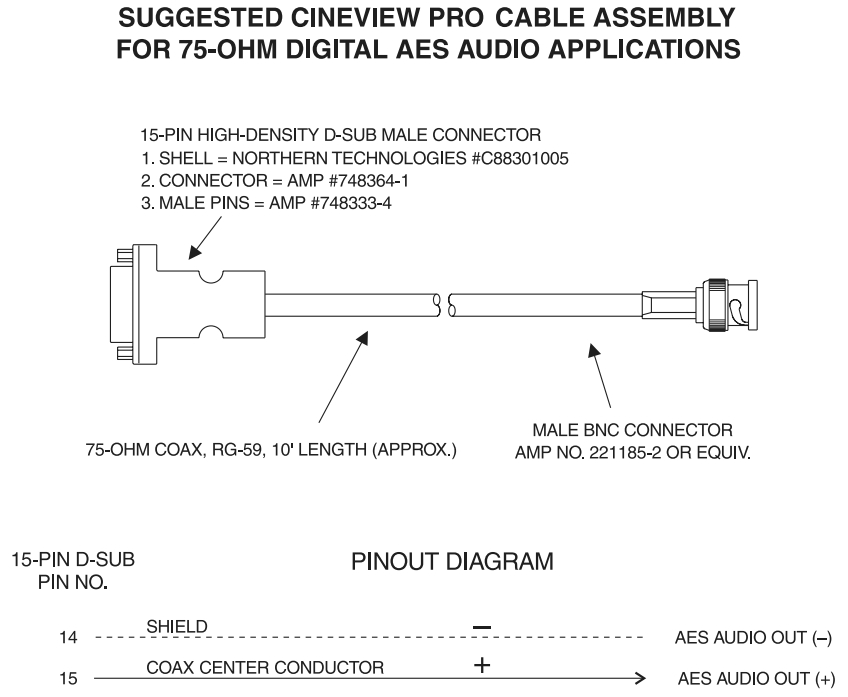


Figure 1-33. Suggested Audio Cabling (Digital 75-Ohm), CineView Pro & XL

Chapter 2

The Playback Application

Introduction

The CineView Pro XL, Pro and Pro LE playback application provides for basic operation of the decoder, including the ability to play, stop, pause and resume MPEG video clips. See Chapter 1 for installation instructions. The source code for the playback application is available for developers desiring to write custom applications. As an aid to programmers, an Application Programming Interface (API) Developer's Guide is included in Part Two of this user manual.

Output Window

Figure 2-1 illustrates the VGA output window and toolbar of the CineView Pro playback application. The window can be toggled on/off through the VGA Output button on the toolbar. The window and toolbar can each be moved to any convenient location by clicking on the title bar and dragging to the desired area. See Figure 2-2 for a detailed view of the playback application toolbar.

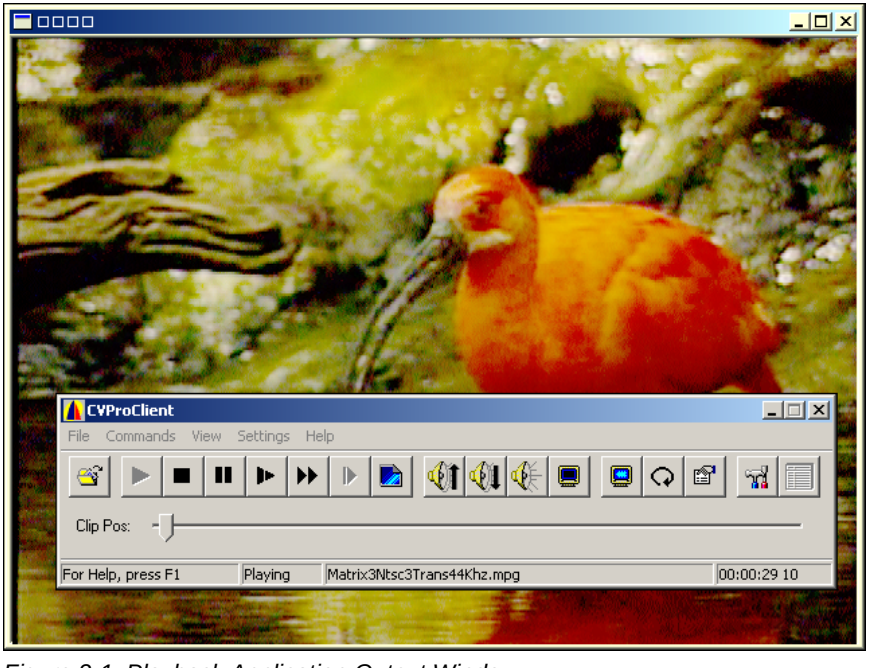


Figure 2-1. Playback Application Output Window

Playback Toolbar

The playback toolbar appears each time the playback application is started. Toolbar button objects are described below. A tool tip is displayed for each active button object as the mouse cursor passes over it. In addition, a menu bar (an item-by-item description of which can be found on page 48), is available at the top of the toolbar. The menu bar's drop-down windows duplicate many of the functions of the toolbar buttons. The status bar, located at the bottom of the playback toolbar window, displays real-time information on the decoder and current clip playback. For detailed information, see "Status Bar," page 47.

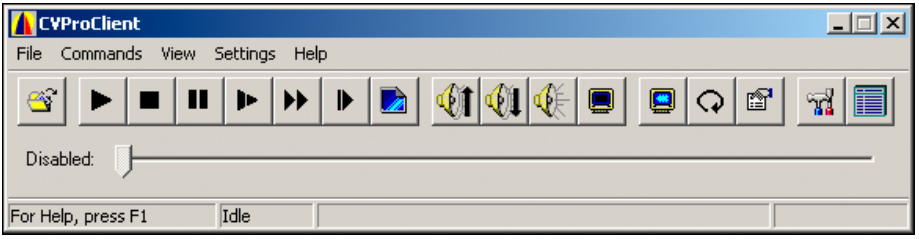


Figure 2-2. Playback Application Toolbar

Open – Click to open a browser window for the selection of an MPEG file for playback. See Figure 2-3 for a sample window. To begin playback, click on a desired file name, then press the Play button. The application recognizes and will play back files with the following file extensions:

.mpg (System, program, and transport streams)

.pcm (PCM audio)

.vbs (Video elementary and PES)

.vpl (Playlist)

Play – Click to start playback of the MPEG clip. (This button is also used to resume from Pause, Slow Motion, Fast Forward and Freeze Frame.)

Stop – Click to stop playback of the MPEG video clip.

Pause – Click to pause playback of the clip. Click the Play button to resume normal playback of the clip. The Default is to pause on the individual video field, unless the "Freeze Frame" box is checked in the General setup window (Figure 2-11).

Slow Motion – Click to play the video in slow motion. Click the Play button to resume normal playback.

Fast Forward – Click to play the clip in fast motion. Click the Play button to resume normal playback of the clip.

Frame Advance – Advances video one frame at a time.

Freeze Frame (or Field) – Click to hold and view the current video frame or field *while the MPEG clip continues to decode*. Click the Play button to resume normal playback. Default is to freeze the individual video field, unless "Freeze Frame" is checked in the General setup window (Figure 2-11).

Audio Volume Up – Click to increase audio volume in 2dB increments. The icon will be grayed out when the upper volume limit is reached.

Audio Volume Down – Click to decrease audio volume in 2dB decrements. The icon will be grayed out when the lower volume limit is reached.

Mute/Unmute − Click to mute or restore the audio portion of the clip playback.

Video Blanking – Click to blank out the audio and video on both the host computer and through the external video and audio outputs. Click again to restore the signal.

VGA Output – Click to enable or disable VGA display of the current clip playback. Does not affect external video.

Loop Playback – Click to toggle on/off continuous playback of the current video clip.

Properties – Click to access a window containing stream information about the currently-open MPEG file. See Figure 2-5 for a sample properties window.

Setup – Click to access the CineView Pro setup windows, which is displayed by pressing the desired tab atop the window:

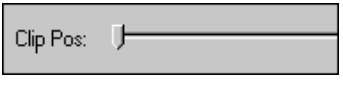
General playback settings (Figure 2-11).

VGA display settings (Figure 2-12).

Audio/Video output settings (Figure 2-13).

On Screen Display settings (Figure 2-14).

Playlist – Click to bring up the Playlist GUI window. Playlist allows the user to create, modify and/or play back a list of MPEG files. See “Using Playlist,” page 51.



Mid-Stream Start Slider – This slider bar allows the operator to start video playback as far into the video stream as he/she desires without having to start at the beginning of the clip. Simply click on the slider and move it to the desired location relative to the length of the clip.

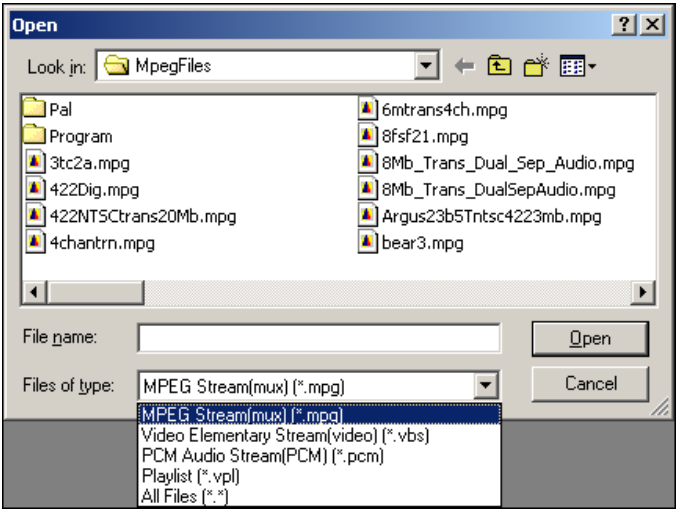


Figure 2-3. File Open Window with Supported Types

Status Bar

As shown in Figure 2-4, the status bar, a subdivided area making up the lower border of the playback toolbar, can display real-time information on the decoder and current playback session.

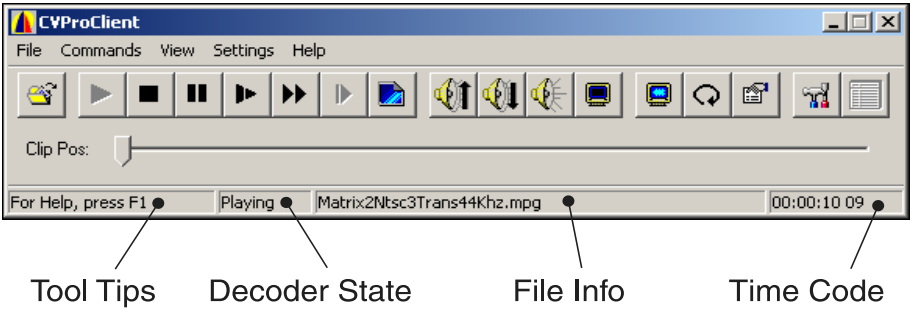


Figure 2-4. Status Bar Layout

Tool Tips – Displays helpful information on the current tool or function in use.

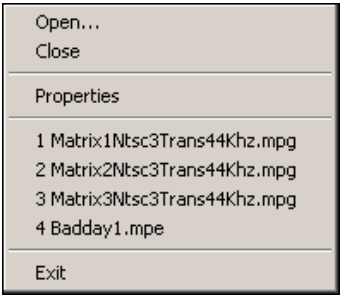
Decoder State – Displays the current state of the decoder.

File Info – Displays the file name of the current MPEG clip (or the file name of the last clip played).

Time Code – Displays GOP time code as read by the on-board DSP processor from the MPEG video stream.

Menu Bar

Items along the menu bar at the top of the window can be clicked on to bring down sub-menus that duplicate many of the functions of the toolbar buttons.



File Menu – Displays a drop-down window with several functions:

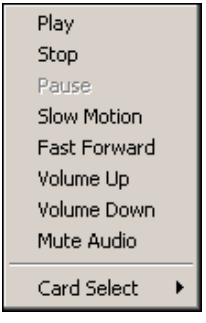
• Click **Open** to bring up a browser window (Figure 2-3), from which an MPEG file is selected for playback.

• Click **Close** to close the file currently in use.

• Click **Properties** to bring up the MPEG File Properties window, (Figure 2-5), providing stream information and other data on the file that is currently open.

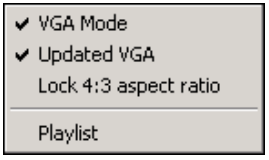
• The four most recently opened files are shown for user convenience in reopening them. Selecting one of the displayed file names will result in immediate playback of the selected clip.

• The playback application is terminated by clicking on **Exit**.



Commands Menu – A few often-used playback operations can be found here, including **Play**, **Stop**, **Pause**, **Slow Motion**, **Fast Forward**, and audio volume and mute controls.

• **Card Select** allows the user to select the decoder card desired for file playback. Up to four decoders can be installed.



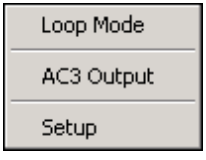
View Menu – Shows available display options:

VGA Mode: Enables or disables VGA display of the current clip playback. Does not affect external video.

Updated VGA: Enables the use of an updated VGA display that allows windows to be played on top without the picture showing through. Full-screen playback is also possible when Updated VGA is selected.

Lock 4:3 aspect ratio: Prevents inadvertent resizing of the VGA output window width/height ratio by locking in a 4:3 aspect ratio. Has no effect on external video.

Playlist: Opens the Playlist window, allowing the user to create or modify clip playlists.

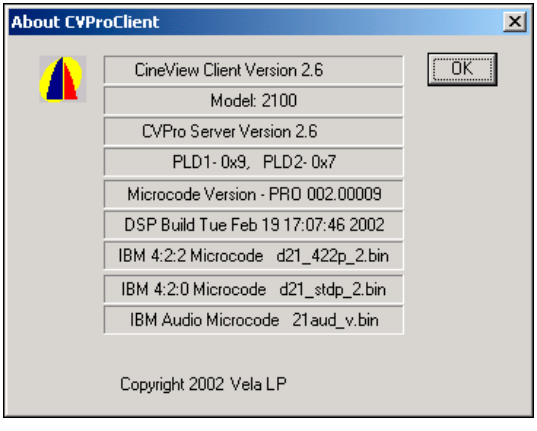


Settings Menu – Shows the following options:

Loop Mode: Click for continuous playback of a clip.

AC-3 Output: Click to enable output of an AC-3 output bit stream. This will also disable one audio channel.

Setup: Click to open the Setup window (Figure 2-11), allowing the adjustment of VGA properties, audio/video parameters, OSD settings, etc.



Help Menu – Clicking on Help > About will display various data including the current software version, build number, firmware revision (EPLD code), and microcode version of the installed decoder. IBM decoder microcode for audio and 4:2:2/ 4:2:0 video versions are also listed in Help > About.

MPEG File Properties Window

(Figure 2-5) This window displays properties of the MPEG file that is currently selected for viewing. It is accessed from either File Menu > Properties or from the Properties button on the application toolbar.

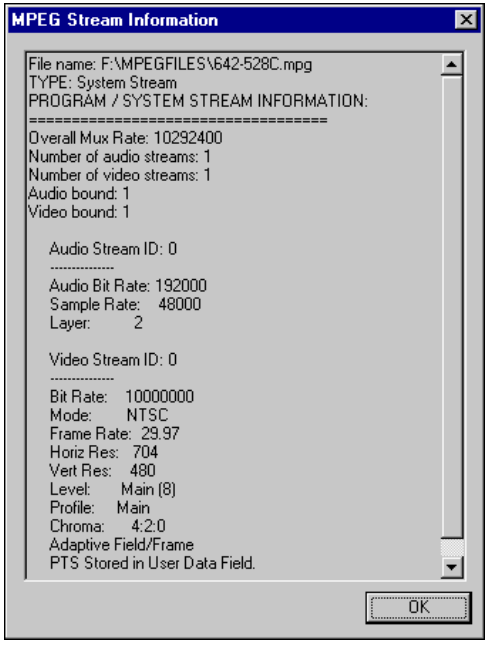


Figure 2-5. MPEG File Properties Window

Using Playlist

The CineView Pro Playlist feature is used to create, modify, and/or play back a list of files in the playlist. In order to do this, activate the Playlist GUI by clicking on the Playlist button on the playback toolbar or click on Playlist in the drop-down View menu (Figure 2-6). For an explanation of buttons and features on the toolbar, refer to “Playback Toolbar,” page 44.

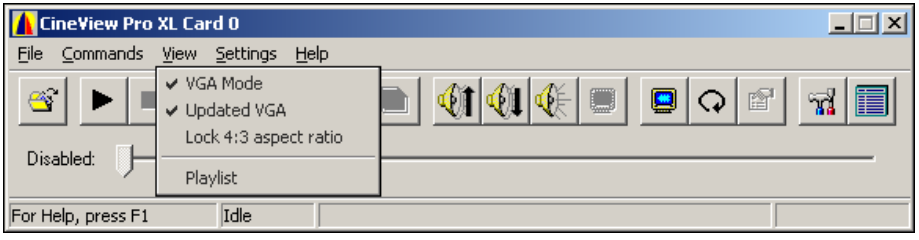


Figure 2-6. Playback Toolbar with View Menu Dropdown

Creating, Modifying, or Opening a Playlist

The Playlist GUI window (Figure 2-7) is used to create new playlists or modify existing ones. To create a playlist, with the .vpl file extension, MPEG files are added to the list. This is done by clicking and dragging files into the list area of the GUI or by clicking on the Add button. When the Add button is clicked, an Open File window will appear (Figure 2-8). Use this window to select the clip file(s) to be added to the playlist.

To open an existing playlist, click on the Playlist GUI window Browse button (labeled “…”). This will bring up an Open Files dialog window (Figure 2-9) from which a playlist file can be selected and opened. Please note that only one playlist file can be opened and played at a time.

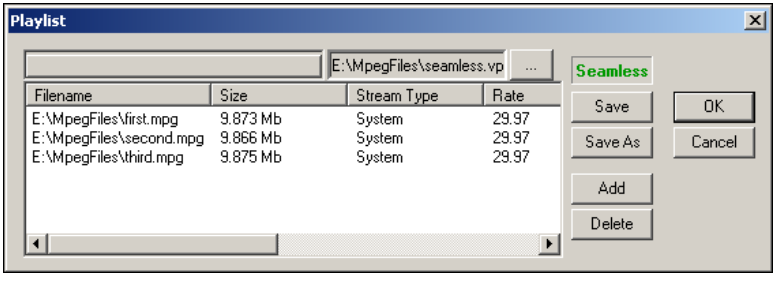


Figure 2-7. Playlist GUI Window

Once a playlist file is opened, it can be modified. New clips can be added to it by clicking on the Add button. Clips can be deleted by clicking on the Delete button. The order of the playback of the clips can be changed. This is done by clicking and dragging a clip to a desired position in the list.

"Seamless" Indicator – This indicator, located on the Playlist GUI window (Figure 2-7), shows whether or not a playlist is seamless. "Seamless in red means that the playback will have black between clips in the playlist. "Seamless" in green means that the playback will not have black between clips. In order for the playlist to be seamless, the clips must have same stream type and frame rate.

Saving a Playlist File

There are three ways of saving a playlist (.vpl) file. The first way is to click on the Save button on the Playlist GUI window. This will save the list under a current playlist name. However, if there is no playlist name specified, a Save As window (Figure 2-10) will appear to assist in saving the list with a name. Another way to save a playlist file is to click on the Playlist GUI Save As button. This works the same was as the Save button when no playlist name is present. The third way of saving a playlist is to click on the OK button. This acts just like the Save button, except it will exit the Playlist GUI after the playlist is saved.

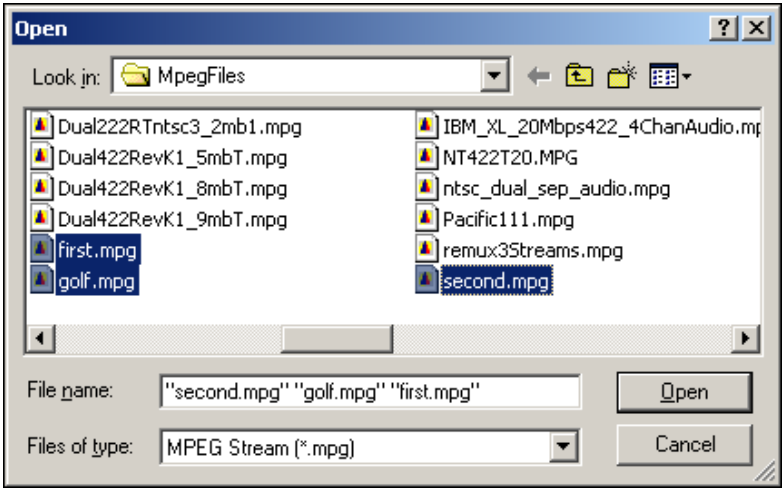


Figure 2-8. MPEG Clip "Open Files" Window

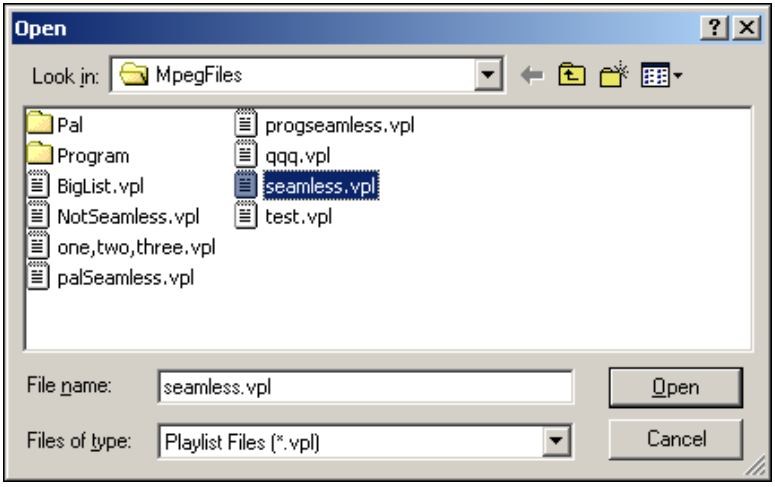


Figure 2-9. Playlist "Open Files" Window

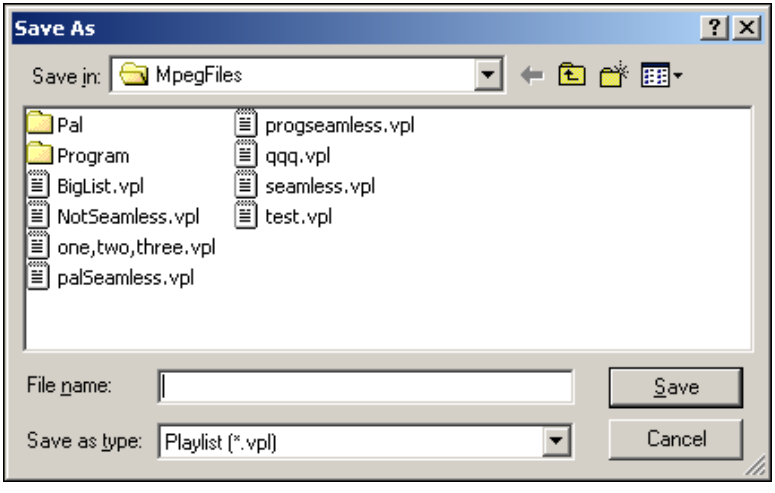


Figure 2-10. Playlist "Save As" Window

Playback Setup Windows

When the Setup button is clicked on the main playback application toolbar, or the Settings menu Setup selection is clicked, one of three system setup windows will appear. To switch among the windows, click the tabs displayed in each window.

General Setup Window

(Figure 2-11) This window is where parameters related to overall video clip playback are set and changed.

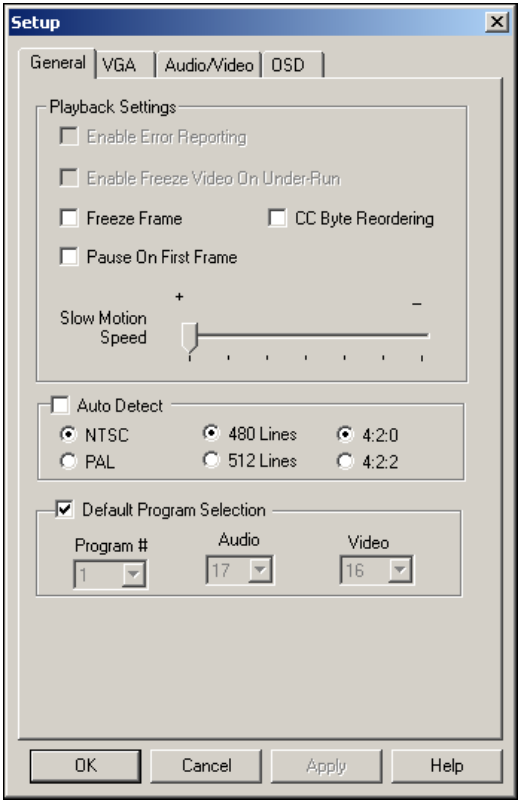


Figure 2-11. General Setup Window

Enable Error Reporting – This feature is under development and is not currently supported.

Enable Freeze Video on Under Run – This feature is under development and is not currently supported.

Freeze Frame – When checked, video will automatically freeze on the first video frame of the selected playback clip when the file is opened. This can serve as a convenient cueing or pre-loading tool. To start playback, press Play as normal. Default is unchecked.

CC Byte Reordering – This box, when checked, allows proper display of ATSC and other byte-reordered closed caption streams. The default setting is unchecked.

Pause on First Frame – Check this box if the Pause and Freeze Frame modes are to display a frame of video. When unchecked (default), a field of video is displayed rather than a video frame.

Slow Motion Speed – Set the slider to the speed desired for video playback in slow motion. This can be done while the video is playing.

Auto Detect – Check this box if auto-detection of the playback format is desired. Otherwise, click the appropriate radio button for type playback appropriate for the selected clip. Box is checked by default.

NOTE: NTSC and PAL mode changes are applied as their respective radio buttons are clicked. The number of video lines and output format selections are applied only when the OK or Apply buttons are clicked. Those settings are then permanently stored in the Windows Registry.

Default Program Selection – Check this box if a multi-program transport stream is in use. The list boxes are then used to select the particular clip desired and its audio and video settings. Transport streams are the only type currently supported under this feature. Box is checked by default.

OK Button – Click to save the current settings to the on-board EEPROM and dismiss the window.

Cancel Button – Click to cancel the new settings, return to current values and dismiss the window.

Apply Button – Click this button to save the current settings to the on-board EEPROM. The window will not be dismissed.

Help Button – Click to obtain help on this window.

VGA Setup Window

(Figure 2-12) This window is where changes are made that affect the appearance of the host computer VGA display. The values set here have no effect on the composite or serial D-1 external outputs.

NOTE: Editing any value on this window causes real-time changes to be applied to the current video playback. By clicking on the OK and/or Apply buttons, the changes made will be made permanent and stored in the on-board EEPROM rewriteable chip.

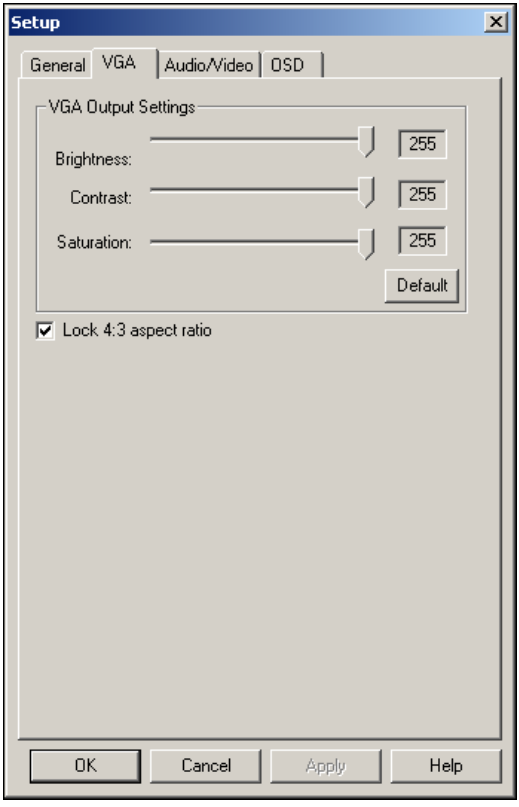


Figure 2-12. VGA Setup Window

Brightness – Adjusts the display screen image brightness level.

Contrast – Adjusts the display screen image contrast level.

Saturation – Adjusts the display screen image color saturation.

Default Button – Restores brightness, contrast, and saturation levels to their default settings.

Lock 4:3 aspect ratio – When checked, prevents inadvertent resizing of the VGA output window width/height ratio by locking in the broadcast standard 4:3 aspect ratio. Has no effect on external video.

OK Button – Click to save the current settings to the on-board EEPROM and dismiss the window.

Cancel Button – Click to cancel the adjustment procedure and return to current values. The window will be dismissed.

Apply Button – Click this button to save the current settings to the on-board EEPROM. The window will not be dismissed.

Help Button – Click to obtain help on this window.

Audio/Video Setup Window

(Figure 2-13) This window displays current video and audio playback parameters, and allows changes to those parameters.

NOTE: Editing any value on this window causes real-time changes to be applied to the current video playback. By clicking on the OK and/or Apply buttons, the changes made will be made permanent and stored in the on-board EEPROM rewritable chip.

Video Gain – Adjusts the video signal level to the proper amplitude.

Black Level – Sets reference level of picture black (“setup”). The default setting corresponds to the broadcast standard value of 7.5 IRE units.

Blanking Level – Sets reference level of the video blanking signal. The default setting corresponds to a value of 0 (zero) IRE units.

1st Active Line – First line of active video. Usually set to 7, but some encoders place first line at 6. Default setting is 7. Applies to 512-line encodes only.

Enable 0 IRE Setup – Check this box if zero-IRE (“enhanced black level”) setup is required. This is sometimes necessary because, while the NTSC system sets video black at 7.5 IRE units, the Japanese NTSC standard and PAL format countries set black at 0 IRE units. Box is checked by default.

Gain U – Adjusts the chroma amplitude along the U axis.

Gain V – Adjusts the chroma amplitude along the V axis.

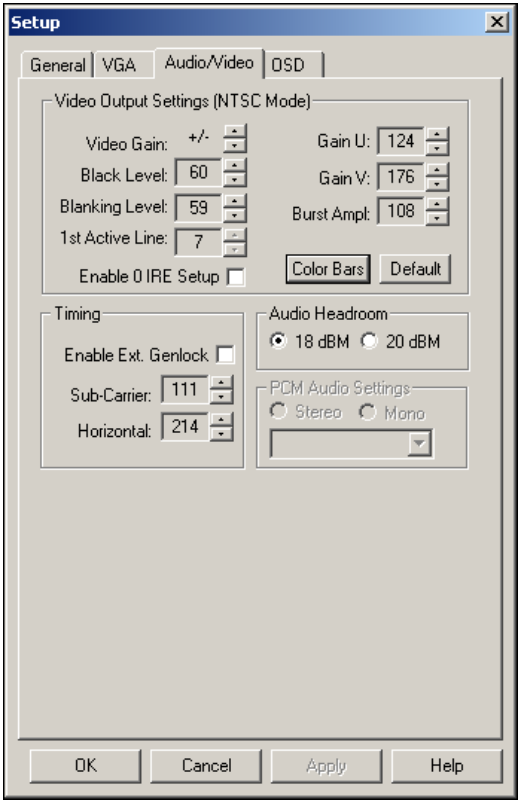


Figure 2-13. Audio/Video Setup Window

Burst Ampl – Sets the amplitude of the chroma burst signal.

Color Bars Button – Click to bring up a color bar display. For use with composite video output only.

Default Button – Click to restore all settings on this window, except Video Gain, to factory defaults.

Enable Ext. Genlock – Check this box if the output video signal is to be locked to an external video source. Note that the desired external source video must be connected to the genlock connector on the CineView Pro decoder board.

Sub-Carrier – Adjusts the output burst phase relative to the genlock input.

Horizontal – Adjusts the output sync phase relative to the genlock input.

Audio Headroom – Click the radio button for 18dBm or 20dBm as desired to match the audio system used for the CineView Pro decoder.

PCM Audio Settings – This feature is under development and is not currently supported.

OK Button – Click to save the current settings to the on-board EEPROM and dismiss the window.

Cancel Button – Click to cancel the new settings, return to current values and dismiss the window.

Apply Button – Click this button to save the current settings to the on-board EEPROM. The window will not be dismissed.

Help Button – Click to obtain help on this window.

On Screen Display Setup Window

(Figure 2-14) The OSD Setup Window is where the On Screen Display parameters are set and OSD control takes place.

Make Decoder Ready for OSD – Check this box to inform the application that the current decoder board is ready to receive OSD commands. If this box is not checked, everything within this group will be disabled. Also, this box cannot be checked at the same time that either AutoDetect or Pause on First Frame boxes are checked in the General Setup Window.

NOTE: Checking this box does not mean that OSD is enabled.

Bitmap File – This edit box shows the name of the bitmap file that is to be loaded for OSD. New bitmaps can be chosen by typing the bitmap file name in the edit box or by clicking on the Browse (…) button. When only the bitmap file name is typed in, the application looks for this file in the directory that was previously set in the Registry. Therefore, if the new bitmap file is in another directory, the drive name and the folder must also be typed, i.e.”c:\bitmapfiles\bitmapfile.bmp.” Note that the backslash “\” and the “.bmp” extension must be used where appropriate.

Position X – Position X identifies the horizontal position of the upper left corner of the OSD region. This number has to be between 0 and 700 inclusive. Each number represents one pixel on a horizontal line.

Position Y – Position Y identifies the vertical position of the upper left corner of the OSD region. Unlike in Position X, here each number is represented by two

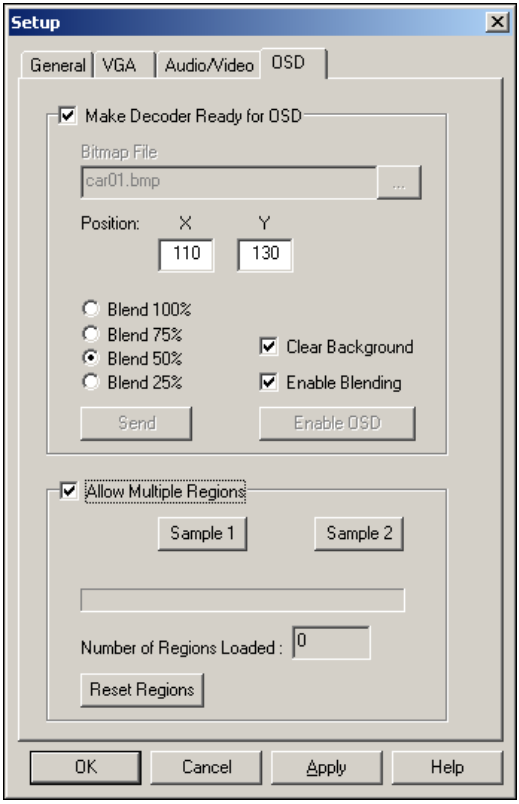


Figure 2-14. OSD Setup Window

vertical pixels. Therefore, depending on which Stream Type and Lines are chosen in the General Setup Window, the Position Y constriction is applied by dividing the line chosen by two.

Blend 100%, 75%, 50%, and 25% – These four radio buttons represent the blending (transparency) level for the OSD region. If Blend 100% is checked, there will not be any transparency and no video shows through the OSD region. If Blend 75% is checked, 25% of the video shows through the OSD region. If Blend 50% is checked, 50% of the video shows through the OSD region. And if Blend 25% is checked, 75% of the video shows through the OSD region.

NOTE: Each time the blending level is changed, Send button must be clicked again in order to load the bitmap according to the new blending changes.

Clear Background – If this option is enabled, the first pixel color of the bitmap loaded for OSD is assumed to be the background color and that color will be cleared when OSD is loaded. If this option is changed, the Send button must be clicked again in order to load the bitmap accordingly.

NOTE: If the bitmap has some foreground colors that are the same as the background color, those colors in the foreground will also be cleared when OSD is loaded.

Enable Blending – If this option is checked, the blending level radio buttons will be enabled allowing the control of the blending level. When this option is not checked, the radio buttons will be disabled and the blending level is set to 100%, which shows no transparency. When this option is changed, OSD gets disabled, the Enable/Disable OSD button gets disabled, and the Send button gets enabled. The Send button must be clicked again in order to load the bitmap according to the new blending mode.

Send – When this button is clicked, the decoder gets initialized and the bitmap file gets loaded for OSD. This is when the settings such as, bitmap file directory, bitmap file name, x position, y position, blending level, clear background, and enable blending are saved to the Registry. Therefore, canceling will not have an effect once the send button is clicked.

NOTE: In order to load bitmap, the decoder must be in idle state. Thus the send command will not work while a clip is playing.

Enable/Disable OSD – This button enables and disables OSD. This can be done while a clip is playing.

Allow Multiple Region – This option tells the application whether or not it is ready to load multiple regions. It allows up to 16 bitmaps to be loaded for OSD, and those bitmaps can be used for animation by enabling each one at synchronized time intervals. When this box is checked, everything but positioning, blending levels, clearing background, and blending in the “Make Decoder Ready for OSD” group will be disabled, and everything in the “Allow Multiple Regions” group will be enabled. Any bitmaps loaded will be reset. OSD will be disabled when this box is being checked and unchecked.

Sample1 – This button loads 16 bitmaps for animation. It loads 16 car figures and slides downward at about -45 degrees. This sample takes the X and Y position from the X and Y edit boxes in the “Make Decoder Ready for OSD” group for the first bitmap being loaded. The remaining bitmaps are incremented within the application. X is incremented by 10 pixels and Y is incremented by six pixels in order to simulate a moving car.

Sample2 – This button loads nine bitmaps for animation. It loads nine Vela logos that have comets orbiting. This sample takes the X and Y position from the X and Y edit boxes in the “Make Decoder Ready for OSD” group and sets the same position for all bitmaps.

NOTE: When one of the two sample buttons is clicked, it takes couple of seconds for the application to load all the bitmaps. When the number in the “Number of Regions Loaded” box changes, or when the progress bar is filled, the application is finished loading the bitmaps. When multiple bitmaps are being loaded, the blending, blending levels, and clearing background settings are taken from the “Make Decoder Ready for OSD” group and sets those same settings for all bitmaps.

Progress Bar – Shows the progress of bitmaps being loaded for OSD.

Number of Regions Loaded – This edit box shows how many bitmaps are loaded for OSD.

Reset Regions – This button resets to zero the number of regions loaded and disables OSD. Once this button is clicked, one of the sample buttons must be clicked again in order to display OSD animation.

OK Button – Click to save the current settings to the on-board EEPROM and dismiss the window.

Cancel Button – Click to cancel the new settings, return to current values and dismiss the window.

Apply Button – Click this button to save the current settings to the on-board EEPROM. The window will not be dismissed.

Help Button – Click to obtain help on this window.

Part Two

API Development

Chapter 3

API Overview

Section Overview

This portion of the manual, Part Two, is intended to facilitate the development of custom user interfaces to the CineView Pro XL, CineView Pro and CineView Pro LE decoder boards using the Vela Application Programming Interface and the associated Software Developer's Kit (SDK). It is assumed that the developer has a working knowledge of Microsoft Visual C++, COM, and programming in a Windows 2000 or NT environment.

NOTE: Vela's CineView Pro SDK covers the complete family of CineView Pro decoders (CineView Pro XL, Pro and Pro LE), and will be generally referred to in this section as the CineView Pro API (or SDK).

CineView Pro Decoder Family Details

CineView Pro XL Decoder

The Vela Model 2000-2153 CineView Pro XL decoder is designed to play back MPEG-1 and MPEG-2 main-level main-profile system, transport, and program streams. Digital and composite video is output through respective BNC connectors on the rear of the decoder board. Composite video can also be played back through the system VGA monitor for convenient monitoring. An additional BNC on the rear of the decoder allows a genlock signal to be applied. The CineView Pro XL decoder can output either serial D-1 video or NTSC/PAL composite video. Longitudinal Time Code also can be output.

Pro XL audio output capabilities include analog (+4dBm = 0Vu) and digital (AES/EBU) audio is output via a high-density 15-pin D-sub connector. Digital audio can also be embedded in the SDI stream. S/P DIF audio is also available.

The CineView Pro XL receives MPEG data over the PCI bus. The data is buffered through a 64kbyte FIFO prior to being demuxed by the on board DSP. The TMS320C32 is responsible for demuxing, performing DMA's, loading all microcode, initializing the audio and video decoders, controlling lip sync functions, frame-accurate starts and stops, and general house keeping.

The video decoder is the IBM CD21. The output of the CD21 is sent through a PLD/FPGA to the Gennum SDI interface, the SAA7185 video encoder, the transcoding connector and the VIP connector. The video PLD/FPGA is responsible for adding SAV/EAV codes and switching between different sources and

destinations for the output video. The optional video FPGA will provide for audio embedding of up to 4channels and an LTC output.

The Audio decoders are the Crystal CS4924 AC-3/Musicam decoders. The outputs of the decoders are sent to the CS8420 sample rate converters with built in AES transmitters. The bi-phase outputs of the sample rate converters are directly connected to the AES transformers. The I2S outputs are connected to the video FPGA, the Audio DAC's(PCM1716) and the transcoding connector.

CineView Pro and Pro LE Decoders

The CineView Pro and Pro LE decoders are designed to play back MPEG-1 and MPEG-2 main-level main-profile system, transport, and program streams. Composite video can be played back through the system VGA monitor and is available to an external monitor or video system through a standard BNC connector. Additional BNCs found on each of the decoders allow a genlock signal to be applied.

Exclusive to the Vela Model 2000-2100 CineView Pro decoder, serial D-1 video can be output (through a third BNC connector). Audio, either analog (+4dBm = 0Vu) or digital (AES/EBU) is output via a high-density 15-pin D-sub connector. Digital video and audio output is available only on the CineView Pro decoder.

The Model 2000-2110 CineView Pro LE decoder supports analog audio and video output. See the illustrations at the end of Chapter 1 for board layout views of the decoders.

Both decoders use a Philips® PCI Multimedia Bridge chip with a built-in scaler as its interface to the PCI bus. The PCI bridge chip allows real-time video to be routed to any physical memory location accessible by the PCI bus. This gives the user the ability to route video to the VGA controller. The real-time video can be formatted into a variety of output formats for extended capability with existing VGA frame buffers. The video image size is controlled by a scaling unit located on the decoder. The internal clipping unit allows for window overlapping to be handled easily. Communication to the board's IBM® decoder chip is done through the Data Expansion Bus Interface (DEBI) port of the PCI chip.

Minimum System Requirements

- Microsoft Windows 2000 or Windows NT 4.0 (NT Service Pack 6a).
- IBM® PC or PC-compatible Pentium® 200 system (or higher) with PCI bus.
- 32MB RAM (64MB recommended).
- CD-ROM drive (for installation of system files).

- Vela CineView Pro XL, CineView Pro, or CineView Pro LE decoder board.
- PCI-compliant VGA controller with accessible linear frame buffer (support for high bit 16 or high bit 24 required).
- Fast/Wide SCSI hard drive (recommended) or EIDE hard drive (EIDE limited to max. 6Mbps MPEG-2 files on playback).

Software Requirements

- Vela CineView Pro Software Developer's Kit.
- C++ compiler (Microsoft Visual C++ 6.0 or later running in a 32-bit programming environment).
- Microsoft Visual Basic 6.0 or later (for COM samples).
- Text editing application (for editing initialization files).

Files Included in the SDK

- Sample Visual C++ applications.
- Complete source code for the sample applications.
- Philips SAA7146 drivers.
- Type libraries, .dll files, and COM components:

  SAA46_32.dll and corresponding **SAA46_32.lib**

  CVProApiDLLU.dll

  46denc32.dll and **46denc32.lib**

  46dmsd32.dll and **46dmsd32.lib**

  CineViewServer.tlb

  CVProServer.exe

  FileParseU.dll

  VFCU.dll

  MemMgrServer.exe

  MemoryManager.dll

  Vela_Pins.dll

Installation

Refer to Chapter 1 of this manual for complete hardware, software, and SDK installation instructions for the particular CineView Pro decoder you are using.

Building An Application SDK

- Use the .tlb file to access the public interfaces of the classes.
- Add the following file to the project:

  ddraw.lib
- Include the following conditional compile flags:

  _AFXDLL **_MBCS**
  _UNICODE **_WIN32_DCOM**
  _WINDOWS **NDEBUG**
  WIN32
- For the CineView Pro family of decoders, use the following classes: *CVelaArbCntrl, CVelaDebi, CVelaIo, CVelaGlobalBuffer,* and *CVelaVideoStream.*

Recommendations for Project Settings

- Use MFC in a shared DLL.
- Optimizations: Use maximum speed.
- Pointer to member representation: Always best-case.
- Processor: **Blend***
- Calling convention: **_cdecl**
- Use runtime library: Multi-threaded DLL.
- Struct member alignment: 4 bytes.
- Inline function expansion: **Only_inline**
- Precompiled headers: Not using.
- Preprocessor definitions:

  _AFXDLL **_MBCS**
  _UNICODE **_WIN32_DCOM**
  _WINDOWS **NDEBUG**
  WIN32

- Object library module:

  ddraw.lib

  Resource preprocessor definitions: **NDEBUG** and **_AFXDLL**

Suggested Reading

The following are some suggested reading materials. If the developer has not been exposed to COM or C++ programming, it is highly recommended that a short course be taken in the applicable subject.

C++ and Visual C++ Textbooks

The C++ Programming Language, Stroustrup. The C++ bible. If you do not have this book and are going to be doing any C++ programming, you should make it a point to own it.

Using Visual C++ 5, Gregory, QUE Publishing. Extensive reference for Microsoft's VC++ 6.0.

Visual C++ 4 How-To, Stanfield, Waite Group Press. Valuable book. Covers many areas of MFC and VC++ programming in general. Many GUI how to's.

MFC Internals, Shepherd and Wingo, Addison-Wesley. A good book on the inner workings of MFC.

Microsoft Foundation Class Library Programming, Holzner, Brady Books. Another fine MFC programming reference.

Books on Component Object Model (COM)

Inside COM, Rogerson, Microsoft Press. Recommended book for general COM introduction. Covers COM programming explicitly from a C/C++ hard core, low level mode. Very detailed.

Professional DCOM Programming, Dr. Richard Grimes, WROX Publishing. Chapters 1 and 2 are excellent overviews of COM and other methods of component programming.

ActiveX Controls Inside Out, Denning, Microsoft Press. This book covers a lot of the specifics behind COM concepts.

Essential COM, Box, Addison-Wesley. An excellent book, explains the concepts and workings of COM.

Understanding ActiveX and OLE: A Guide for Developers & Managers, Chappel, Microsoft Press. A high level book that explains the reasoning behind COM.

Customer Support

In the event of questions or problems with Vela Application Programming Interface materials or this manual, do not hesitate to contact Vela Training and Support.

- Phone — (727) 507-5301
- E-mail — support@vela.com
- World Wide Web — http://www.vela.com

Chapter 4

The COM API

The COM-based API for the Vela CineView Pro family of decoders provides a set of binary-independent methods that can be used on any Microsoft Windows 2000 or NT-based development system that supports ActiveX and COM (Component Object Model). For example, the methods can be included in user interfaces written in Microsoft Visual C++™ or Microsoft Visual Basic™. They can even be used with Microsoft end-user applications such as MS Access™. Refer to Chapter 5 for detailed descriptions of many of these methods and properties.

Before a component can be used, appropriate entries must be created for it in the Windows Registry database. The COM component that comprises the CineView Pro API contains self-registration code, so all that is needed is for an application to point to the component in the Registry.

NOTE: This CineView Pro COM API covers the CineView Pro XL, the CineView Pro and the CineView Pro LE decoders, and will be subsequently referred to in this section as the CineView Pro API.

CVProServer Class Methods

The methods for the *CVProServer* class distributed with this API include:

Method	Description
CloseStream	Closes VGA video stream.
CVLock	Locks the interface for exclusive use by one client.
CVUnlock	Unlocks the interface.
FastForward	Changes playback mode to fast forward.
FrameAdvance	Advances the paused video by one frame.
FreezeFrame	Freezes on a single frame while the slider continues to scroll through the MPEG data at normal playback speed.
GetAudioAttenuation	Gets the current audio volume level.
GetAvailStreamRate	Gets the available stream rate.
GetBitsPerPixel	Gets the bits per pixel for a video format.

Table 4-1. CVProServer *Class Methods*

Method	Description (Continued)
GetBitsMoved	Gets variable tracking number of bytes.
GetBytesMoved	Returns the total number of bytes sent to the decoder since playback began.
GetCurrentSettings	Gets the current video setup parameters.
GetDecoderState	Returns the current state of the decoder
GetDecoderVersion	Reads board version id from SAA7146.
GetDefaultSettings	Retrieves the default settings from the tables.
GetLockID	Gets the ID of the current client that has locked the interface.
GetMasterReg	Gets PCI register contents.
SetMasterReg	Writes value to a specified PCI register.
GetMicrocodeVersion	Gets the microcode build version from the DSP.
GetNextPlayListFile	Gets next playlist filename.
GetStreamInfo	Returns file parse results including bit rate, muxed rate, frame rate, chroma format etc.
GetStreamState	Gets the current state of the video stream.
GetTimeCode	Returns current displayed frame time code.
Initialize	Loads client data for the specified PCI device.
Initialize2	Necessary initialization for multiple clients.
Mute Audio	Mutes audio output from CineView Pro.
OpenStream	Opens VGA video stream.
OpenStreamUni	Opens VGA video stream (Unicode).
OpenStream2	Opens VGA video stream.

Table 4-1. CVProServer *Class Methods (Continued)*

Method	Description (Continued)
OpenStream2Uni	Opens VGA video stream (Unicode).
Pause	Pauses the playback.
Play	Begins MPEG file playback.
Play2	Begins playback of an MPEG data stream.
Play3	Begins MPEG file playback from a byte offset.
PlayFromPin	Begins playback of data from a pin.
PlayPcmAudio	Begins PCM file playback from a byte offset.
PlayPcmAudio2	Begins PCM file playback from a byte offset.
ReadEpldRevision	Returns programmable hardware revision.
Reset	Stops the playback.
ResetBytesMoved	Resets variable tracking number of bytes.
ResetVideoReference	Writes the video amplitude reference value to CineView Pro EEPROM.
Resume	Resumes playback following a Pause, Freeze Frame, Slow Motion, or Fast Forward.
SetAudioAttenuation	Sets the audio volume level.
SetClipMask	Allows the video stream to use a custom clip mask.
SetClipMaskUni	Allows the video stream to use a custom clip mask (Unicode).
SetDestRect	Sets the destination rectangle.
SetDestRectUni	Sets the destination rectangle (Unicode).
SetFilePos	Begins file playback at specified byte offset.
SetMidStreamStart	Sets a flag indicating playback to begin at a non-zero byte offset.

Table 4-1. CVProServer *Class Methods (Continued)*

Method	Description (Continued)
SetNextPlayListFile	Sets the next playlist filename for use after the current file playback completes.
SetSrcRect	Sets the source rectangle.
SetSrcRectUni	Sets the source rectangle (Unicode).
SetVisibleRect	Sets video window-visible rectangles.
SlowMotion	Changes playback mode to slow motion.
StartStream	Starts streaming video to VGA display.
StoreVideoSettings	Stores current video settings in EEPROM.
Stop	Stops the playback.
StopStream	Stops streaming VGA video.

Table 4-1. CVProServer *Class Methods (Continued)*

OnScreenDisplay Class Methods

The methods for the *OnScreenDisplay* class distributed with this API include:

Method	Description
EnableOSD	Enables or disables the OSD loaded in the LoadBitmap function.
Initialize	Initializes the current decoder to receive OSD commands.
LoadBitmap	Loads the bitmap file to be used for OSD.
LoadRegion	This method has not been fully implemented.

Table 4-2. OnScreeDisplay *Class Methods*

AVStream Class Methods

The methods for the *AVStream* class distributed with this API include:

Method	Description
Close	
Open	
Start	
Stop	

Table 4-3. AVStream *Class Methods*

FrameControlledPlayback Class Methods

The methods for the *FrameControlledPlayback* class distributed with this API include:

Method	Description
FrameAccuratePlay	Starts frame-accurate playback of video file.
Initialize	Prepares frame-accurate interface to play back video file.

Table 4-4. FrameControlledPlayback *Class Methods*

Component Overview

The CineView Pro API is not really a set of function calls. The traditional C++ style API simply does not apply in the world of COM. Instead, we look at components and component functionality. A CineView Pro component must be able to Start, Stop, Pause, and Resume its data flow. To be fully functional, the component must be able to present its settings to the user in a graphical manner. And of course, the component must present its settings to the API user in a programmatic manner.

All Vela components, as with other COM objects, present to the user methods and properties. At this point, it is suggested that any C programmers reading this guide have on hand a good C++ reference book or two. Programmers who haven't yet ventured into the world of COM or OLE /Active-X may want to pick up a book or two and do some studying of those subjects. A list of suggested reading materials can be found near the end of Chapter 1.

A method is simply a public interface. The method will do something and will return a result code of some sort. To work well with COM, and with the future expansion into DCOM, every method call in our components returns an HRESULT return code. This return code should always be checked for validity and success of the function. The HRESULT is a 32-bit integer. Visual Basic will automatically intercept the error and go to an exception handler. A method is exposed to the user through a type library.

A property can be thought of as a C++ public data member within a class. For non-C++ programmers, a property is simply a variable located within the component that the application programmer can access. All of the settings in the components are accessible as properties. Some properties are read/write while others are read only. For example, the PauseOnFirstFrame mode is a boolean value that the user can test to (a) determine if the decoder is in PauseOnFirst-Frame mode or (b) put the decoder in PauseOnFirstFrame mode if it isn't.

COM places strict rules on the developer of COM objects. The foremost rule is that an interface (properties and methods) will always work the same way. Vela components are designed to adhere to that rule. For example, the Pause method will always work as designed regardless of the version of the component. If we were to need a Pause type of function that took more parameters or performed a slightly different function, we would develop a Pause2 method, instead of change Pause, and support both functions in the next release.

The API developer is guaranteed that, once an application is developed using the CineView Pro decoder API, future releases of COM components will not affect the operation of the application. Of course, to use the new features, new code will need to be written and recompiled.

Some of the properties presented here might change usage in the future. For example, some versions of the API might no longer use a specific property. This does not mean that Vela will no longer support the property. It will still be available, but the component might ignore that property in a specific release.

What follows is a brief explanation of the groups of common method calls. Specifics for properties and other filter specific methods are outlined in the sections to follow.

Processing Methods

The methods of CineView Pro components return an **HRESULT** type value representing the status of the decoder or the success of the call. The functionality of these methods is defined earlier in this chapter.

CVProServer Class Properties

A number of properties are available to the application developer to customize the behavior of the CineView Pro decoder. Currently supported *CVProServer* properties are as follows:

Property	Description
AC3Mode	BOOL read/write property to indicate if AC-3 bit stream output is enabled or disabled.
Audio20dbfs	Short read/write property that determines the audio headroom.
AudioPID	Short read/write property that commands DSP to set up decoder to play audio program ID with the specified value.
AutoDetect	BOOL read/write property to indicate if the source of video settings is file parse or manual entries from the checkboxes.
Bars	BOOL read/write property indicating decoder mode to output color bars.
BlackLevel	Integer read/write property that determines the video black level setting.
BlankLevel	Integer read/write property that determines the video blank level setting.
BlankVideo	BOOL write property that determines if there is output of video to the VGA window.
Brightness	Integer read/write property that determines the brightness level of the video.
BurstAmplitude	Integer read/write property that determines the video burst amplitude setting.
ByteReorderMode	BOOL read/write property that determines the closed captioning byte order.
ChromaPhase	Integer read/write property indicating the video chroma phase setting.

Table 4-5. CVProServer *Class Properties*

Property	Description (Continued)
ChromaType	BOOL read/write property indicating 4:2:0 or 4:2:2 chroma format setting.
Contrast	Integer read/write property that determines the contrast level of the video.
FirstVideoLine	Integer read/write property indicating whether the first video appears on line 6 or line 7.
FreezeFieldMode	BOOL read/write property indicating whether to pause on a single field or a frame.
GainU	Integer read/write property that determines the video U-Axis gain setting.
GainV	Integer read/write property that determines the video V-Axis gain setting.
GenLock	BOOL read/write property indicating whether the decoder is in genlock mode.
HorizontalPhase	Integer read/write property indicating the video horizontal phase setting.
MaxLineResolution	BOOL read/write property indicating to the 7185 DENC device to operate in low or high resolution mode and becomes part of every DSP Play command.
MuxedStreamType	Integer read/write property that determines the stream type of the muxed data.
NTSC	BOOL read/write property indicating NTSC or PAL mode.
NumDecoders	Short read property that determines the number of decoder cards in the system.
PauseOnFFMode	BOOL read/write property indicating whether decoder is in pause on first frame mode.
PlayListMode	BOOL read/write property indicating whether decoder is in playlist mode, with no black.

Table 4-5. CVProServer *Class Properties (Continued)*

Property	Description (Continued)
ProgramID	Integer read/write indicating the program ID number for playback.
Saturation	Integer read/write property that determines the saturation level of the video.
ShowVGADisplay	BOOL read/write property that determines whether or not to show the updated VGA display.
SlowMotionRate	Integer read/write property that determines the speed of the slow motion playback.
VGADisplay	BOOL read/write property indicating which VGA display is in use. TRUE for updated VGA; FALSE for old.
VideoPID	Short read/write property that commands DSP to set up decoder to play a video program ID with the specified value.
VideoReference	Integer read/write property indicating the reference video amplitude.
ZeroIRE	BOOL read/write property that set the black level to 0 IRE units or 7.5 IRE units and rein-initializes the 7185 hardware.

Table 4-5. CVProServer *Class Properties (Continued)*

AVStream Class Properties

Currently supported *AVStream* properties are as follows:

Property	Description
AudioOutPin	String
VideoOutPin	String

Table 4-6. AVStream *Class Properties*

FrameControlledPlayback Class Properties

Currently supported *FrameControlledPlayback* properties are as follows:

Property	Description
EndControlType	Specifies which property the FrameControlledPlayback interface will use to end the playback.
EndFrame	Sets frame number of the frame at which the playback will stop.
EndTime	Sets the time code at which the playback will stop.
FrameControlMode	Not Implemented.
FrameDuration	Specifies the number of frames from the start point at which the playback will stop.
PauseOnFF	Property is set to TRUE to enable pausing on the first frame.
PauseOnLF	Property is set to TRUE to enable pausing on the last frame.
PreBlack	Not implemented.
StartControlType	Specifies which property the FrameControlledPlayback interface will use to start the playback.
StartFrame	Sets frame number of the frame at which the playback will start.
StartTime	Sets the time code at which the playback will start.

Table 4-7. FrameControlledPlayback *Class Properties*

Events

Currently there is one event that can be sent, which is the windows message:

VELA_PLAYLIST_ITEM_COMPLETE – This windows message is sent by the COM object to the client, indicating that the current Playlist file has completed. The client should respond by calling SetNextPlayListFile, if there are more files in the playlist to play. If the playlist has completed, then the API will interpret an empty filename to mean the end of playlist.

In order to capture this message, you must initialize the CVProServer COM

object using the Initialize2() method. The second parameter contains a handle of the window to which the message will be sent. You must add code to capture this windows message (refer to the example code on the CD-ROM).

Constants

There are a number of decoder state and stream type constants available to the application developer to facilitate development.

Decoder State Constants

STOP = 0x00

PLAY = 0x01

PLAY_PCM = 0x02

PAUSE = 0x03

FASTFORWARD = 0x04

SLOWMOTION = 0x05

FREEZEFRAME = 0x06

FRAMEADVANCE = 0x07

RESYNC = 0x08

Stream Type Constants

AUDIO_ELEM = 0x00

VIDEO_ELEM = 0x01

AUDIO_PES = 0x02

VIDEO_PES = 0x03

MUXED = 0x04

STINVALID = 0x05

Using Vela Components from Visual C++

It is assumed that the developer using these components is or will be familiar with Microsoft Visual C++ (VC++) 6.0. While it is entirely possible to access and use these components from other compiler packages, no examples are given. Other packages (such as Borland or Watcom) should behave in a similar manner.

The VC++ 6.0 workplace makes using COM components easy. Within VC++, the developer can define an ATL or MFC-based project. When creating a project, it is very important that you select OLE Automation from the App Wizard form.

If you are adding COM support to an existing project, or if your project does not use MFC, we highly recommend studying some of the books available on MFC

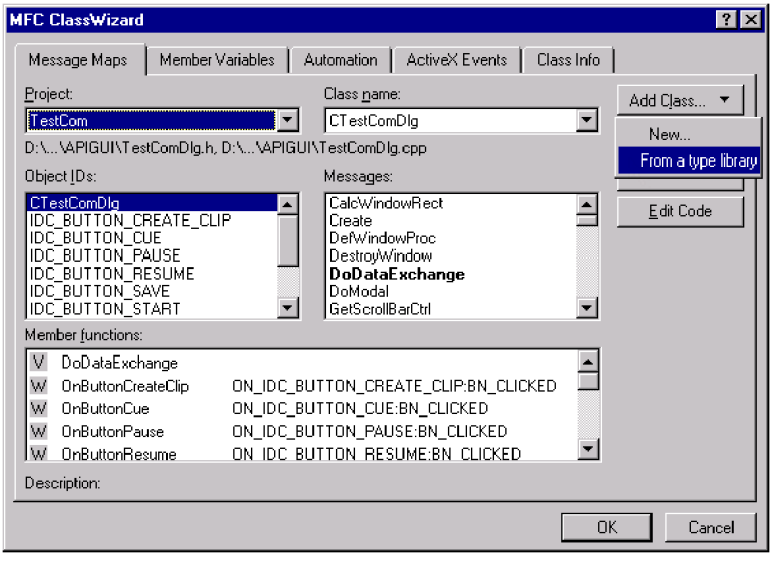


Figure 4-1. VC++ Class Wizard Main Window

core details and COM specifics. We also suggest using MFC as a DLL from App Wizard, as all of the COM components are already using this DLL.

(Figure 4-1) From within your project, open the Class Wizard, and press the Add Class button. You will see a couple of selections. Pick the one labeled “From a Type Library.” Then select the DLL that contains the component you want to use. Click on OK and exit the wizard. Now you have a class in your project that gives you access to the COM component (Figure 4-2).

If you take a moment and examine the class that the wizard created for you, you will notice that there are only functions. What about the properties? If you look closer you will notice that there are a number of Get… and Set… method calls in the class. These are all methods that allow you to set and get the COM properties from within VC++.

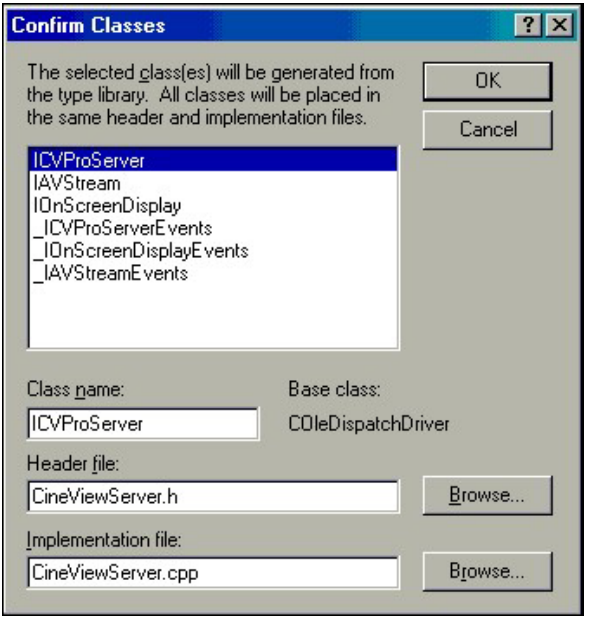


Figure 4-2. Class Wizard Class Creation

However, you can't use the component quite yet. In the introduction we mentioned the fact that the component had to be registered with the operating system. We can now use that registration information from within our program. Since the COM component is registered with the operating system, these Win32 function calls will query the file system to see if the component exists. If so, we can grab a handle to the component, which we can then type cast into the C++ class that the Class Wizard created for us (Figure 4-3).

Before we are able to actually use any COM calls, we must first initialize COM in our code. This is done with a call to CoInitialize(), passing the function a NULL parameter. It should be noted that in older documentation, COM can also be initialized with a call to AfxOleInit(), OleInitialize(), or other documented functions. The CoInitialize() call may be called multiple times in an application, but it must be matched with a call to CoUninitialize(). Please refer to your Win32 documentation for a more detailed description of these function calls. Code Listing No. 1 shows an example of the steps needed to instantiate a COM component for use inside your application.

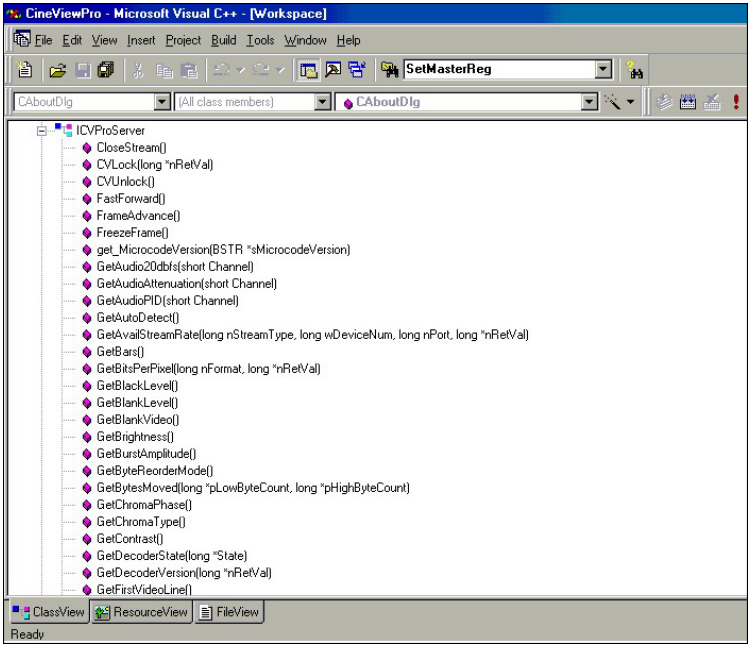


Figure 4-3. Resulting C++ Class in Workspace

At this point in the program we can access the class method calls that were created for us by the Class Wizard. Of course, good programming habits are assumed here, as you should always check for return values (hr, for example). Details about COM and Windows programming are too in-depth for this API document, but should be investigated further by the reader.

Using Vela Components from Visual Basic

Since the CineView Pro API was created using COM methodology, it also has the advantage of being language independent. Because of this, the API can be accessed by just about any programming language. Due to the popularity of Visual Basic (VB), this manual also includes the following example for using the API from VB.

This code defines the variable “myCineViewPro” as a CineView Pro object, creates an instance of the object, then accesses the interface methods in a manner similar to VC++. Notice that “myCineViewPro” is Dim'd “WithEvents.” This is done so the object can receive events. See the documentation on VB for additional information. The major difference between C++ and VB access to the object is how the properties are presented to the user. In C++, the properties are

Code Listing #1 (Visual C++)

```
ICineView *m_pCineView;
_Module.Init(NULL, NULL);

hr=CoInitialize();
// Create instance of the CineView object
 hr=CoCreateInstance(CLSID_CineView,NULL,
CLSCTX_INPROC_SERVER,
    IID_ICineView, (LPVOID*) &m_pCineView);
...
// initialize decoder
hr = m_pCineView->Initialize(&result);
...
// Create instance of event interface
CComObject<CCineViewEvent>* pCineViewEvent=NULL;
 CComObject<CCineViewEvent>::CreateInstance(&pCineViewEvent);
 m_dwCookie = 99;
 hr = AtlAdvise( m_pCineView, pCineViewEvent->GetUnknown(),
    IID_ICineViewEvents, &m_dwCookie);
...
AtlUnadvise(m_pCineView, IID_ICineViewEvents, m_dwCookie);
...
CoUnitialize();
```

accessed via Get… and Set… methods. From VB, the properties appear as simple variables attached to the filter object. For example, setting the LoopMode property would look like this: CineViewPro.LoopMode = True.

Developing a CineView Pro Application

The CineView Pro API is the heart and soul of the application development process. This section will cover the development of an application using the CineView Pro API. Using the basic concepts outlined here will give the developer a choice of development paths that can not be found with any other decoder system on the market today. If you have not performed the API installation, please do so now. Two sample applications are included. One is written in Visual C++ 6.0 using MFC and ATL, and can be built and run with the decoder component that is delivered with the CineView Pro decoder board. A second application is written in Visual Basic 6.0.

Code Listing #2 (Visual Basic)

```
Public WithEvents myCineView As CINEVIEWSERVERLib.CVProServer

Sub Form_Load()
    Set myCineView = new CINEVIEWSERVERLib.CVProServer
    myCineView.Initialize 0 ‘ Initializes Card number 0
End Sub

Sub Play_Click()
Dim strCurrentFile as String
Dim lRetVal as Long
Dim lStreamType as Long
    lStreamType = 4 ‘ Multiplexed stream
    strCurrentFile = “D:\mpegfiles\first.mpg”
    myCineView.Play strCurrentFile, lStreamType, lRetVal
End Sub

Sub Form_Unload()
    Set myCineView = Nothing
End Sub
```

What is the purpose of the CineView Pro API? The main function of the API is to provide a layer of abstraction between the application GUI and the decoder. This was done for two reasons:

1. Object Abstraction. This gives Vela flexibility when shipping different board sets or different system configurations in the future. Our application will remain exactly the same, and will communicate with an API that creates and uses the appropriate components for that decoder. This lowers our testing cycle to a minimum, and allows for future expansion. Furthermore, the application developer is able to quickly change the GUI to meet specific requirements.

2. Encapsulation. The low level interaction between the hardware and software is encapsulated within a single API. This allows for changes to the hardware or software localized to a single COM object.

The application performs the following three roles:

1. Interacts with the decoder through a standard API.

2. Includes a toolbar and pull-down menus for the end-user to manipulate.

3. Provides a dialog to modify the brightness, contrast, and saturation of the VGA output.

If you examine the C++ code in the sample application provided, you will see sample code which uses the three roles performed by the application. The toolbar and pull-down menus can be found in **mainfrm.cpp**. The code to manipulate the VGA settings can be found in the **VGADlg.cpp**. You are free to use the application as a base for your development process. The idea of the CineView Pro API is that it can be expanded beyond that of a simple decoder. You can integrate the sample application with other applications, or you may simply use the CineView Pro API without using any GUI.

Chapter 5

Advanced API Development

The following descriptions and usage examples of COM-based methods and properties are presented here for the benefit of advanced developers who can use the extra information in their CineView Pro program development efforts. An introductory look at the CineView Pro API methods and properties can be found in Chapter 4.

The term “CineView Pro” is used here to represent the entire CineView Pro family of decoders: CineView Pro XL, CineView Pro, and CineView Pro LE.

CVProServer Class Method Descriptions

VSReturn **CloseStream**

Description

This method instructs the decoder card to stop sending video data to its designated output, normally a VGA screen on the monitor.

Return Value

VSReturn is an enumeration. Its definition and values can be found in the “Definitions” section of this chapter, which starts on page 195.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     rValue as VSReturn
.
.
.
DoEvents
' Allow any windows messages to be processed before continuing.
rValue = objCVProServer.CloseStream()
' stops the decoder card from sending video data
```

Example (VC++)

```
ICVProServerPtr         m_pService;
VSReturn                vsValue;
.
.
.
vsValue = m_pService->CloseStream();
// stops the decoder card from sending video data
```

Void **CVLock** (long nRetVal)

Description

This method is an interlock mechanism of the COM Server which insures that only one client at a time is allowed to change critical parameters. If the decoder is not currently playing, or if no other client has obtained the lock, then this method returns LOCK_AQUIRED, indicating success. If the lock is not acquired, it may return a CURRENTLY_LOCKED or CURRENTLY_PLAYING status. A client must have the lock in order to play or to modify critical parameters.

Parameters

long nRetVal — A variable of type long that will store the result of the call to attempt to lock the CVPro COM object. The return value can be as follows:

0 = LOCK_AQUIRED
1 = CURRENTLY_LOCKED
2 = CURRENTLY_PLAYING

Return Value:

None

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     nRetVal as Long
objCVProServer.Initialize(0)
' Initializes card number zero. Any other commands will affect card 0 only.
.
.
objCVProServer.CVLock(nRetVal)
' Attempts to lock the COM object. The variable nRetVal contains the result
' of the attempt
```

Example (VC++)

```
ICVProServerPtr          m_pService;
long                     nRetVal;
m_pService->Initialize(0);
// Initializes card number zero. Any other commands will affect card 0 only.
.
.
m_pService->CVLock(nRetVal);
// Attempts to lock the COM object. The variable nRetVal contains the result
// of the lock attempt.
```

Void **CVUnlock**

Description

This method releases the interlock mechanism, providing the requesting client is the client that obtained the lock. This method always returns an S_OK status.

Parameters

None

Return Value

None

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     nRetVal as Long
objCVProServer.Initialize(0)
' Initializes card number zero. Any other commands will affect card 0 only.
.
objCVProServer.CVLock(nRetVal)
' Application attempts to lock the COM object. The variable nRetVal contains the result
' of the attempt.
.
objCVProServer.CVUnLock()
' Application unlocks the COM object.
```

Example (VC++)

```
ICVProServerPtr          m_pService;
long                          nRetVal;
m_pService->Initialize(0);
// Initializes card number zero. Any other commands will affect card 0 only.
.
.
m_pService->CVLock(nRetVal);
// Application attempts to lock the COM object. The variable nRetVal contains the result
' of the lock attempt.
.
m_pService->CVUnLock();     // Application unlocks the COM object.
```

DEBIReturn **FastForward**

Description

This method instructs the decoder card to skip frames to give the appearance of moving forward at a quicker pace.

Return Value

DEBIReturn is an enumeration. Its definition and values can be found in the "Definitions" section of this chapter, which starts on page 195.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lRetVal as Long
objCVProServer.Initialize(0)
.
.
.
lRetVal = objCVProServer.FastForward
```

Example (VC++)

```
ICVProServerPtr             m_pService;
int                             m_nRetVal;
m_nRetVal = m_pService->FastForward();
```

DEBIReturn **FrameAdvance**

Description

This method instructs the decoder board to advance by a single video frame.

Parameters

None

Return Value

DEBIReturn is an enumeration. Its definition and values can be found in the "Definitions" section of this chapter, which starts on page 195.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lRetVal as Long
objCVProServer.Initialize(0)
.
.
.
lRetVal = objCVProServer.FrameAdvance
```

Example (VC++)

```
ICVProServerPtr          m_pService;
int                      m_nRetVal;
m_nRetVal = m_pService->FrameAdvance();
```

DEBIReturn **get_MicrocodeVersion** (BSTR *sMicrocodeVersion)

Description

This method instructs the decoder card to return the microcode version installed on the CineView Pro card.

Parameter

BSTR sMicrocodeVersion: A string variable that will store the result of the query to the CineView Pro card.

Return Value

DEBIReturn is an enumeration. Its definition and values can be found in the "Definitions" section of this chapter, which starts on page 195.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     strVersion as String
Dim     lRetVal as Long
objCVProServer.Initialize(0)
.
.
.
lRetVal = objCVProServer.get_MicrocodeVersion(strVersion)
```

Example (VC++)

```
ICVProServerPtr          m_pService;
BSTR                         VersionStuff;
int                              nReturnCode;
nReturnCode = m_pFrame->m_pService->get_MicrocodeVersion(&VersionStuff);
```

Long **GetAudioAttenuation**

Description

This method returns the audio attenuation (volume) level that the CineView Pro card is set to.

Parameters

None

Return Value

The returned attenuation value of type long can range from 0 to 63, inclusive. A “0” means the card is outputting at its highest volume and “63” means the card is outputting at its lowest volume.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lRetVal as Long
objCVProServer.Initialize(0)
.
.
.
lRetVal = objCVProServer.GetAudioAttenuation
' lRetVal will contain a value between 0 and 63 inclusive
```

Example (VC++)

```
ICVProServerPtr          m_pService;
Short                    nValue;
int                      nChannel;
int                      m_nAttenuation;
m_nAttenuation = m_pService->GetAudioAttenuation(nChannel);
m_pService->SetAudioAttenuation(nChannel, nValue);
```

void **GetAvailStreamRate** (long StreamType, long wDeviceNum, long nPort, int *nRetVal)

Description

Not currently implemented. Always returns the “error not supported” status specified below.

Parameters

long StreamType

long wDeviceNum

long nPort

int *nRetVal

Return Value

VS_ERR_NOT_SUPPORTED

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lStreamType as Long, lDeviceNum as Long, lPort as Long, lRetVal as Long
' Indicate stream type is muxed type.
lStreamType = 4
' Indicate device and port number
lDeviceNum = 0
lPort = 0
' Call function.
objCVProServer.GetAvailStreamRate(lStreamType, lDeviceNum, lPort, lRetVal)
```

Example (VC++)

```
ICVProServerPtr          m_pService;
long                         nStreamType;
long                         nDeviceNum;
long                         nPort;
int                          nRetVal; // stream rate return value
m_pService->GetAvailStreamRate(nStreamType,wDeviceNum,nPort,&nRetVal);
```

void **GetBitsPerPixel** (int nFormat, int *nRetVal)

Description

This method returns the number of bits per pixel for a given video format. An enumeration that indicates a particular video format.

Parameters

VideoFormat nFormat. The definition for this enumeration can be found in the Definitions section at the end of this chapter.

Return Value

An integer value indicating the number of bits per pixel.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lFormat as Long, lRetVal as Long
lFormat = 1
' 3 bytes per pixel, R8, G8, B8
objCvProServer.GetBitsPerPixel(lFormat, lRetVal)
' lRetVal should contain the value 24.
```

Example (VC++)

```
ICVProServerPtr          m_pService;
int                      nFormat;
int                      nRetVal;
m_pService ->GetBitsPerPixel(nFormat, &nRetVal);
```

void **GetBytesMoved** (long *pLowByteCount, long *pHighByteCount)

Description

This method returns the total number of bytes sent to the decoder since playback began. This information may be used to drive a slider bar indicating the current position in a MPEG file.

Parameters

long *pLowCount

long *pHighCount

Return Value

Two pointers to 32-bit integers that together represent the 64-bit total number of bytes sent to the decoder.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lLowCount as Long, lHighCount as Long
objCVProServer.GetBytesMoved (lLowCount, lHighCount)
```

Example (VC++)

```
long                              low, high;
ICVProServerPtr                   m_pService->GetBytesMoved(&Lower, &Upper);
m_pService->GetBytesMoved(&lower, &high);
```

long **GetCurrentSettings** (LPDEFAULTCVPROSETTINGS Settings, long *lRetVal)

Parameters

LPDEFAULTCVPROSETTINGS Settings

long lRetVal

Return Value

A structure containing the following video settings information. For the definition of the LPDEFAULTCVPROSETTINGS structure, see the “Definitions” section of this chapter, which starts on page 195. The RetVal parameter returned is not currently implemented.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     defCurrentSettings as CineViewServer.DefaultCVProSettings
objCvProServer.GetCurrentSettings(defCurrentSettings)
```

Example (VC++)

```
DEFAULTCVPROSETTINGS    pCurrentSettings;
ICVProServerPtr                 m_pService->(&pCurrentSettings);
m_pService->(&pCurrentSettings);
```

void **GetDecoderState** (lState as Long)

Description

This method returns the state of the decoder at the time of the function call.

Parameters

lState: A variable that stores the value that indicates the state of the decoder card. The values that can be passed to this variable are as follows:

0 = Stop (decoder stopped)
1 = Play (decoder playing at normal speed)
2 = Play_PCM (decoder playing PCM audio-only)
3 = Pause (decoder paused)
4 = Fast Forward (decoder in fast forward)
5 = Slow Motion (decoder in slow motion)
6 = Freeze Frame (decoder in freeze frame mode)
7 = FrameAdvance (decoder in frame advance mode)
8 = Resync (decoder has recovered from error and stopped)

Return Value

None

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lState as Long, lRetVal as Long
objCVProServer.Play("C:\mpegfile.mpg", 4, lRetVal)
objCVProServer.GetDecoderState(lState)
' lState has the value of 1
```

Example (VC++)

```
VelaDecoderState        eState;
ICVProServerPtr         m_pService->GetDecoderState((int *)&eState);
m_pService->GetDecoderState((int *)&eState);
```

void **GetDecoderVersion** (long *nRetVal)

Description

This method retrieves the type of decoder card installed in a computer.

Parameters

nRetVal: An integer value representing the type of decoder card installed, as follows:

0x2083 = CineView
0x2100 = CineView Pro
0x2110 = CineView Pro LE
0x2200 = CineView Pro XL

Return Value

None

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lVersion as Long
ObjCvProServer.GetDecoderVersion(lVersion)
```

Example (VC++)

```
ICVProServerPtr          m_pService;
int                      dwVersion;
long                     lRetVal;
m_pService->GetDecoderVersion(&dwVersion);
```

long **GetDefaultSettings** (DefaultCVProSettings CVProSettings)

Description

This method retrieves the default settings used for this decoder card.

Parameters

LPDEFAULTCVPROSETTINGS Settings

long lRetVal

Return Value

A structure containing the following default video settings information. See page 195 for the definition of the LPDEFAULTCVPROSETTINGS structure. The RetVal parameter returned is not currently implemented.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     defSettings as CineViewServer.DefaultCVProSettings
Dim     lRetVal as Long
lRetVal = objCvProServer.GetDefaultSettings(defSettings);
```

Example (VC++)

```
ICVProServerPtr             m_pService;
long                        m_lRetVal;
DefaultCVProSettings vSettings;
m_lRetVal = m_pService->GetDefaultSettings(&vSettings);
```

void **GetLockID** (long *nLockID)

Description

This method returns the Client ID number of the one that last called the method CVLock to gain an exclusive lock to the decoder card.

Parameters

long *nLockID

Return Value

Returns integer nLockID corresponding to the Client that currently “owns” the lock.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lLockID as Long
objCVProServer.GetLockID(lLockID)
```

Example (VC++)

```
ICVProServerPtr             m_pService;
m_lRetVal = m_pService->GetLockID(&ReturnValue);
```

void **GetMasterReg** (long nReg, long dwRetVal)

Description

This method is used to retrieve the value stored in the PCI register of the Philips SAA7146 chip. It is mostly used to retrieve the brightness, contrast and saturation settings of the VGA window.

Parameters

long nReg

long dwRetVal

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lReg as Long, lRetVal as Long
lReg = 112
' Key value to indicate we want VGA color settings
objCVProServer.GetMasterReg(lReg, lRetVal)
' lRetVal contains color settings.
```

Example (VC++)

```
ICVProServerPtr             m_pService;
```

```
long    color;
m_pService->GetMasterReg(VELA_BCS_CTRL, &color);
m_pService->GetMasterReg(nRegister, nRetVal);
```

---

void **GetNextPlayListFile** (BSTR bstrFilename , long llength)

Parameters

BSTR bstrFilename
long llength

Return Value

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
```

Example (VC++)

```
ICVProServerPtr             m_pService;
```

---

long **GetStreamInfo** (BSTR bstrFilename, VelaStreamInfo StreamInfo)

Description

This method allows the developer to retrieve information on a particular stream, indicated by the first parameter, a string that contains a file name of a valid MPEG-2 file.

Parameters

BSTR bstrFilename

VelaStreamInfo

StreamInfo

Return Value

The return value is a pointer to a VelaStreamInfo structure. See the “Definitions” section of this chapter, which starts on page 195, for details.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     objStreamInfo as CineViewServer.VelaStreamInfo
objCVProServer.GetStreamInfo(“D:\mpegfiles\test1.mpg”, objStreamInfo)
```

Example (VC++)

```
ICVProServerPtr             m_pService;
m_pService->GetStreamInfo(m_Filename, &StreamInfo);
```

void **GetStreamState** (long nRetVal)

Description

This method returns the current state of the muxed stream.

Parameters

Return value is a long integer indicating the current Video Stream state.
It can be one of the following values:

0 = VSS_UNDEFINED// Stream is in undefined state

1 = VSS_OPEN// Stream is in open state

2 = VSS_CLOSE// Stream is in close state

3 = VSS_START// Stream is in start state

4 = VSS_STOP// Stream is in stop state

Return Value

None

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lState as Long
objCVProServer.GetStreamState(lState)
```

Example (VC++)

```
ICVProServerPtr             m_pService;
m_pService->GetStreamState(&nState);
```

void **GetTimeCode** (long nFCount, long nRetVal)

Description

This method retrieves the time code of the frame that is being decoded at the time of the call. The value must then be parsed to retrieve the time in hours, minutes, seconds and frames.

Parameters

A long integer indicating the count and a return value for error messages.

Return Value

None.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lFCount as Long, lRetVal as Long
Dim     lHour as Long, lMinute as Long, lSecond as Long, lFrames as Long
objCVProServ.GetTimeCode lFCount, lRetVal
lHour = (lFCount \ (2 ^ 19)) And &H1F
lMinute = (lFCount \ (2 ^ 13)) And &H3F
lSecond = (lFCount \ (2 ^ 6)) And &H3F
lFrames = lFCount And &H3F
```

Example (VC++)

```
ICVProServerPtr                 m_pService;
m_pService->GetTimeCode(&m_lFrameCount, &m_nRetVal);
```

void **Initialize** (long nCardno)

Description

This method initializes a CineView Pro card plugged into the PC.

Parameter

nCardno: A value that indicates the card number in your PC that you wish to initialize and use. Card numbers can range from 0 to 3, inclusive.

Return Value

None

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
objCVProServer.Initialize(0)
' Initializes card 0
```

Example (VC++)

```
ICVProServerPtr                 m_pService;
m_pService->Initialize(0);
```

void **Initialize2** (long nCardno, long hWnd)

Description

This method initializes a CineView Pro card plugged into a PC. This method also ties a Window handle to the COM object to receive Windows messages. This feature is used for playlists of back to back MPEG-2 files with no black frames in between.

Parameters

nCardno: A value that indicates the card number in the PC that is to be initialized. Card numbers can range from 0 to 3, inclusive.

hWnd: A value that represents the handle to a window in the application that will receive Windows messages from our COM object for playlists.

Return Value

None

Example (VB)

```
Dim      objCVProServer as CineViewServer.CVProServer
' In this example, m_edit is an object name for a text box. We will use its handle to
' intercept Windows API messages from the COM object
objCVProServer.Initialize2(0, m_edit.hWnd
```

Example (VC++)

```
ICVProServerPtr                m_pService;
m_pService->Initialize2(0, m_edit->hWnd);
```

void **MuteAudio** (int iChannel, long MuteFlag)

Description

This method will either turn off or turn on the audio playback of an MPEG-2 file, depending on the setting of the MuteFlag value.

Parameters

iChannel: A value that indicates the audio channel number that is to be affected by this method call.

MuteFlag: A value that indicates whether the audio will be turned off or on. Values can be:

0 = Unmute audio
1 = Mute audio

Return Value

None

Example (VB)

```
Dim      objCVProServer as CineViewServer.CVProServer
' The following example will turn off the audio for channel 0.
objCVProServer.MuteAudio(0, 1)
```

Example (VC++)

```
ICVProServerPtr                m_pService;
// The following example will turn on audio for channel 1.
m_pService->MuteAudio(1, 0);
```

void **OpenStream** (VideoPort, VelaPCIPort vPCIPort, long bMinimized, long *nRetVal)

Description

This method sets up the parameters needed to set up a DirectX window to display decoded video data on a computer monitor.

Parameter

VideoPort: A long value that represents the address location of the tVelaVideoPort structure that is filled in by the application. Its definition can be found in the “Definitions” section of this chapter, beginning onpage 195.

VelaPCIPort: A long value that represents the address location of the tVelaPCIPort structure that is filled in by the application. Its definition and values can be found in the “Definitions” section of this chapter, which starts on page 195.

bMinimized: A long value that indicates whether the VGA window should be minimized or not.

nRetVal: A variable of type long that stores the result of the call to the decoder hardware.

Return Value

None

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     objVideo as VelaVideoPort
Dim     objPCI as VelaPCIPort
Dim     lMinimized as Long, lRetVal as Long
' Fill in values of objVideo structure and objPCI structure. Sample values
' can be found in the SDK Visual C++ sample.
.
' VGA window is not to start out minimized.
lMinimized = 0
' Call function
objCVProServer.OpenStream(objVideo, objPCI, lMinimized, lRetVal)
```

Example (VC++)

```
ICVProServerPtr         m_pService;
VelaVideoPort           m_VideoPort;// video port descr. enum
VelaPciPort             m_PciPort;// PCI port descr. Enum
long                        lMinimized, lRetVal;
```

```
// Fill in values of objVideo structure and objPCI structure. Sample values
// can be found in the SDK Visual C++ sample.
.
// VGA window is to start out minimized
lMinimized = 1;
// Call function
m_pService->OpenStream(&m_VideoPort, &m_PciPort, lMinimized, &lRetVal);
```

void **OpenStream2** (VideoPort, VelaPCIPort vPCIPort, long *nRetVal)

Description

This method sets up the parameters needed to set up a DirectX window to display decoded video data on a computer monitor. This method assumes the window that will contain the video output will be displayed normally.

Parameters

VideoPort: A long value that represents the address location of the tVelaVideoPort structure that is filled in by the application. Its definition can be found in the "Definitions" section of this chapter, beginning on page 195.

VelaPCIPort: A long value that represents the address location of the tVelaPCIPort structure that is filled in by the application.

nRetVal: A variable of type long that stores the result of the call to the decoder hardware.

Return Value

None

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     objVideo as VelaVideoPort
Dim     objPCI as VelaPCIPort
Dim     lRetVal as Long
' Fill in values of objVideo structure and objPCI structure. Sample values
' can be found in the SDK Visual C++ sample.
.
 ' Call function
objCVProServer.OpenStream(objVideo, objPCI, lMinimized, lRetVal)
```

Example (VC++)

```
ICVProServerPtrm_pService;
VelaVideoPort           m_VideoPort;// video port descr. enum
VelaPciPort             m_PciPort;// PCI port descr. Enum
long                        lRetVal;

// Fill in values of objVideo structure and objPCI structure. Sample values
// can be found in the SDK Visual C++ sample.
.
// Call function
m_pService->OpenStream(&m_VideoPort, &m_PciPort, lMinimized, &lRetVal);
```

VSReturn **OpenStream2Uni** (VideoPort, VelaPCIPort vPCIPort, long *nRetVal)

Description

This method executes the same procedure that the method OpenStream2 does. This function is made available for those that wish to include Unicode support in their applications.

Parameters

VideoPort: A long value that represents the address location of the tVelaVideoPort structure that is filled in by the application. Its definition can be found in the "Definitions" section of this chapter, beginning on page 195.

VelaPCIPort: A long value that represents the address location of the tVelaPCIPort structure that is filled in by the application. Its definition and values can be found in the "Definitions" section of this chapter, which starts on page 195.

nRetVal: A variable of type long that stores the result of the call to the decoder hardware.

Return Value

A value that is defined by the VSReturn enumeration. Its definition and values can be found in the "Definitions" section of this chapter, which starts on.page 195

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     objVideo as VelaVideoPort
Dim     objPCI as VelaPCIPort
Dim     lRetVal as Long
' Fill in values of objVideo structure and objPCI structure. Sample values
' can be found in the SDK Visual C++ sample.
.
 ' Call function
objCVProServer.OpenStream2Uni(objVideo, objPCI, lMinimized, lRetVal)
```

Example (VC++)

```
ICVProServerPtrm_pService;
VelaVideoPort           m_VideoPort;// video port descr. enum
VelaPciPort             m_PciPort;// PCI port descr. Enum
long                        lRetVal;

// Fill in values of objVideo structure and objPCI structure. Sample values
```

```
// can be found in the SDK Visual C++ sample.
.
// Call function
m_pService->OpenStream(&m_VideoPort, &m_PciPort, lMinimized, &lRetVal);
```

VSReturn **OpenStream2Uni** (VideoPort, VelaPCIPort vPCIPort, long bMinimized, long *nRetVal)

Description

This method executes the same procedure that the method OpenStream does. This function is made available for those that wish to include Unicode support in their applications.

Parameters

VideoPort: A long value that represents the address location of the tVelaVideoPort structure that is filled in by the application. Its definition can be found in the "Definitions" section of this chapter, beginning on page 195.

VelaPCIPort: A long value that represents the address location of the tVelaPCIPort structure that is filled in by the application. Its definition and values can be found in the "Definitions" section of this chapter, which starts on page 195.

bMinimized: A long value that indicates whether the VGA window should be minimized or not.

nRetVal: A variable of type long that stores the result of the call to the decoder hardware.

Return Value

A value that is defined by the VSReturn enumeration. Its definition and values can be found in the "Definitions" section of this chapter, which starts on page 195.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     objVideo as VelaVideoPort
Dim     objPCI as VelaPCIPort
Dim     lMinimized as Long, lRetVal as Long

' Fill in values of objVideo structure and objPCI structure. Sample values
' can be found in the SDK Visual C++ sample.
.
' VGA window is not to start out minimized.
lMinimized = 0
```

```
' Call function
objCVProServer.OpenStream(objVideo, objPCI, lMinimized, lRetVal)
```

Example (VC++)

```
ICVProServerPtrm_pService;
VelaVideoPort                m_VideoPort;// video port descr. enum
VelaPciPort                  m_PciPort;// PCI port descr. Enum
long                              lMinimized, lRetVal;
// Fill in values of objVideo structure and objPCI structure. Sample values
// can be found in the SDK Visual C++ sample.
.
// VGA window is to start out minimized
lMinimized = 1;
// Call function
m_pService->OpenStream(&m_VideoPort, &m_PciPort, lMinimized, &lRetVal);
```

DEBIReturn **Pause** ()

Description

This method causes the decoder to freeze the video at the frame on display when the command is executed.

Parameters

None

Return Value

DEBIReturn is an enumeration. Its definition can be found in the “Definitions” section of this chapter, beginning on page 195.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lRetVal as Long
.
lRetVal = objCVProServer.Pause()
```

Example (VC++)

```
ICVProServerPtrm_pService;
DEBIReturn                  dbReturn;
dbReturn = m_pService->Pause();
```

void **Play** (char *strFileName, long StreamType, DEBIReturn *nRetVal)

Description

This method instructs the CineView Pro decoder to begin decoding the file indicated in the first parameter of the method.

Parameters

strFileName: A string value that indicates the name of the MPEG-2 file that is going to be decoded using the CineView Pro card.

StreamType: An integer value that indicates the stream type of the MPEG-2 file. The values can be the following:

0 = Audio elementary stream
1 = Video elementary stream
2 = Audio packetized elementary stream
3 = Video packetized elementary stream
4 = Multiplexed stream

nRetVal: A variable that stores a return value from the function call. The value returned is actually from the enumeration DEBIReturn. Its definition can be found in the “Definitions” section of this chapter, beginning on page 195.

Return Value

None

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lRetVal as Long
objCVProServer.Play (“D:\MPEGFiles\test1.mpg”, 4, lRetVal)
```

Example (VC++)

```
BSTR                    cFile = _bstr_t(m_Filename);
int                         m_Filetype;
__int64                 byteoffset
long                    High = (long)(byteoffset >> 32);
long                    Low = (long)(byteoffset & 0xffffffff);
ICVProServerPtr         m_pService;
m_nRetVal = m_pService->Play3(cFile, (int)m_Filetype, &High, &Low);
```

void **Play2** (long StreamType, DEBIReturn *nRetVal)

Description

This method instructs the CineView Pro decoder to begin decoding. MPEG-2 data to be used for decoding has already been fed into an internal buffer. This is a carryover from previous versions. Please use the method **PlayFromPin** to begin decoding from these pins or internal buffers.

Parameters

StreamType: An integer value that indicates the type of stream the MPEG-2 file is. The values can be the following:

0 = Audio elementary stream
1 = Video elementary stream
2 = Audio packetized elementary stream
3 = Video packetized elementary stream
4 = Multiplexed stream

nRetVal: A variable that stores a return value from the function call. The value returned is actually from the enumeration DEBIReturn. Its definition can be found in the “Definitions” section of this chapter, beginning on page 195.

Return Value

None

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lRetVal as Long
' Set decoder card to play a video elementary stream.
objCVProServer.Play2(1, lRetVal)
```

Example (VC++)

```
ICVProServerPtr             m_pService;
int                         m_Filetype;
m_nRetVal = m_pService ->Play2((int)m_Filetype);
```

long **Play3** (char *strFileName, long StreamType, long HiOffset, long LowOffset)

Description

This method instructs the CineView Pro decoder to begin decoding at a point in the MPEG-2 file indicated by the HiOffset and LowOffset parameters.

Parameters

strFileName: A string value that indicates the name of the MPEG-2 file that is going to be decoded using the CineView Pro card.

StreamType: An integer value that indicates the stream type of the MPEG-2 file. The values can be the following:

0 = Audio elementary stream
1 = Video elementary stream
2 = Audio packetized elementary stream
3 = Video packetized elementary stream
4 = Multiplexed stream

HiOffset: A long value that represents the first 32 bits of a 64-bit value that indicates at which point along the MPEG-2 file you wish to start.

LowOffset: A long value that represents the second 32 bits of a 64-bit value that indicates where along the MPEG-2 file you wish to start.

Return Value

The value returned is actually from the enumeration DEBIReturn. Its definition can be found in the “Definitions” section, beginning on page 195.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lHigh as Long, lLow as Long
' Set variables so that application begins at file offset 100000bytes
lHigh = 0
lLow = 100000
objCVProServer.Play3(“D:\MPEGFILES\test1.mpg”, 4, lHigh, lLow)
```

Example (VC++)

```
BSTR                    cFile = _bstr_t(m_Filename);
int                      m_Filetype;
__int64                 byteoffset;
long                    High = (long)(byteoffset >> 32);
long                    Low = (long)(byteoffset & 0xffffffff);
ICVProServerPtr         m_pService;
m_nRetVal = m_pService->Play3(cFile, (int)m_Filetype, &High, &Low);
```

void **PlayFromPin** (char strPinName, long StreamType, long nRetVal)

Description

This method instructs the decoder card to obtain the muxed data from a section of memory (called a pin) instead of from a file name. The application is responsible for feeding the data into the pin.

Parameters

strPinName: The name of the OutPin (memory buffer) that contains MPEG-2 data to be used for decoding.

StreamType: An integer value that indicates the stream type of the MPEG-2 file. The values can be the following:

0 = Audio elementary stream
1 = Video elementary stream
2 = Audio packetized elementary stream
3 = Video packetized elementary stream
4 = Multiplexed stream

nRetVal: A variable that stores a return value from the function call. The value returned is actually from the enumeration DEBIReturn. Its definition can be found in the "Definitions" section of this chapter, beginning on page 195.

Return Value

None

Example (VB)

N/A

Example (VC++)

```
ICVProServerPtr          m_pService;
IOutPinPtr               myOutPin = NULL;
bstr_t                   strLocalPin(_T("SerOutPin"));
myOutPin->PutName(strLocalPin);
myOutPin->PutSize(size*20);
result = myOutPin->InitializeAsProcessPin();
nRetVal = myOutPin->Reset();
m_pService.CreateInstance("CineViewServer.CVProServer.1");
m_pService->Initialize2(0,0);
// loop to prefill 20 buffers
        nRetVal = myOutPin->RequestMemory( &mem, &size );
        //Read 64K block of MPEG data directly into Output pin memory pointer.
        m_nBytes_read = fread((void*)mem, sizeof(char), 64*1024, m_pMPEG_FilePtr);
```

```
// end loop
m_pService->PlayFromPin(strLocalPin, StreamInfo.StreamType, &nRetVal);
// loop until done
        nRetVal = myOutPin->RequestMemory( &mem, &size );
        //Read 64K block of MPEG data directly into Output pin memory pointer
        m_nBytes_read = fread((void*)mem, sizeof(char), 64*1024, m_pMPEG_FilePtr);
// end loop
```

void **PlayPCMAudio** (BSTR strFileName, long HiOffset, long LowOffset, long *nRetVal)

Description

This method instructs the CineView Pro decoder card to begin decoding an MPEG-2 file that is an audio stream only. The decoding will begin at the point in the file indicated by the HiOffset and LowOffset values.

Parameters

strFileName: A string value that indicates the name of the MPEG-2 file that is going to be decoded using the CineView Pro card.

HiOffset: A long value that represents the first 32 bits of a 64-bit value that indicates where along the MPEG-2 audio file you wish to start decoding.

LowOffset: A long value that represents the second 32 bits of a 64-bit value that indicates where along the MPEG-2 audio file you wish to start decoding.

nRetVal: A variable that stores a return value from the function call. The value returned is actually from the enumeration DEBIReturn. Its definition can be found in the “Definitions” section of this chapter, beginning on page 195.

Return Value

None

Example (VB)

N/A

Example (VC++)

```
ICVProServerPtr     m_pService;
BSTR                cFile = _bstr_t(m_Filename);
int                 m_Filetype;
__int64             byteoffset;
long                High = (long)(byteoffset >> 32);
long                Low = (long)(byteoffset & 0xffffffff);
ICVProServerPtr     m_pService;
m_nRetVal = m_pService->PlayPCMAudio(cFile, &High, &Low);
```

void **PlayPCMAudio2** (char *strFileName, long StereoMode, long Rate, long HiOffset, long LowOffset, long *nRetVal)

Description

This method instructs the CineView Pro decoder to begin decoding an MPEG-2 file that is an audio stream only. The decoding will begin at the point in the file indicated by the HiOffset and LowOffset values. It also allows the programmer to select whether or not to play the audio in stereo and what sample rate to use when playing the audio file.

Parameters

strFileName: A string value that indicates the name of the MPEG-2 file that is going to be decoded using the CineView Pro card.

StereoMode: A long value that indicates whether or not to play the audio file in stereo. Values can be:

1 = stereo mode
0 = mono mode

Rate: A long value that represents the sample rate to use for this audio file. The values can be as follows:

0 = 22.05 Kilohertz (Khz)
1 = 24 Khz
2 = 16 Khz
4 = 44.1 Khz
5 = 48 Khz
6 = 32 Khz

HiOffset: A long value that represents the first 32 bits of a 64-bit value that indicates where along the MPEG-2 audio file you wish to start decoding.

LowOffset: A long value that represents the second 32 bits of a 64-bit value that indicates where along the MPEG-2 audio file you wish to start decoding.

nRetVal: A variable that stores a return value from the function call. The value returned is actually from the enumeration DEBIReturn. Its definition can be found in the “Definitions” section of this chapter, beginning on page 195.

Return Value

None

Example (VB)

N/A

Example (VC++)

```
BSTR                cFile = bstr_t(m_Filename);
BOOL                m_bStereoMode;
int                 m_nRate;
__int64             byteoffset;
long                High = (long)(byteoffset >> 32);
long                Low = (long)(byteoffset & 0xffffffff);
ICVProServerPtr     m_pService;
m_nRetVal = m_pService-> PlayPcmAudio2 (cFile, m_bStereoMode, m_nRate, &High,
                                        &Low);
```

void **ReadEpldRevision** (long *nRetVal)

Description

This method retrieves the revision number from the programmable logic devices (referred to as PLDs) of the CineView Pro.

Parameter

nRetVal: A variable of type long that stores the revision number found in the PLDs of the CineView Pro.

Return Value

None

Example (VB)

N/A

Example (VC++)

```
int                 EpldModel;
ICVProServerPtr     m_pService;
m_pService->ReadEpldRevision((int *)&EpldModel);
```

long **Reset** (long lPortNo)

Note

This method is not supported at this time.

void **ResetBytesMoved** ()

Description

This method resets the indicator of how many bytes of MPEG data have been processed already back to zero.

Parameters

None

Return Value

None

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
objCVProServer.ResetBytesMoved()
```

Example (VC++)

```
ICVProServerPtr             m_pService;
m_pService->ResetBytesMoved();
```

void **ResetVideoReference** (long NewVal, long *nRetVal)

Description

This method resets the decoder board video reference output level to the midrange setting.

Parameter

NewVal: A long value that indicates the new video reference midrange setting.

nRetVal: A variable of type long that stores the revision number found in the PLDs of the decoder.

Return Value

None

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lRetVal as Long
' Set new video reference to 100
objCVProServer.ResetVideoReference(100, lRetVal)
```

Example (VC++)

```
ICVProServerPtr             m_pService;
m_pService->ResetVideoReference(50, &m_nRetVal);
```

DEBIReturn **Resume** ()

Description

This method instructs the decoder card to return to normal play of the decoded file. This command should only be executed after a trick mode has been executed (i.e. Pause, Fast Forward, Slow Motion).

Parameters

None

Return Value

The value returned is from the enumeration DEBIReturn. Its definition can be found in the Definitions section of this chapter, which begins on page page 195.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lReturn as Long
lReturn = objCVProServer.Resume()
```

Example (VC++)

```
ICVProServerPtr             m_pService;
m_nRetVal = m_pService->Resume();
```

void **SetAudioAttenuation** (int Channel, int Attenuation)

Description

This method allows the user to set the volume level of the audio output.

Parameters

Channel: An integer value that indicates the channel number that you wish to set the attenuation (volume) on.

Attenuation: An integer value that indicates the level of attenuation the card is outputting. This value can range from 0 to 63, inclusive, where:

0 = highest amount of volume that can be set
63 = lowest amount of volume that can be set.

Return Value

None

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     iChannel as Integer, iAttenuation as Integer
' Set channel to 0 and attenuation to 0
iChannel = 0
iAttenuation = 0
' Set volume
objCVProServer.SetAudioAttenuation (iChannel, iAttenuation)
```

Example (VC++)

```
int                         m_nAttenuation;
ICVProServerPtr             m_pService;
m_pService->SetAudioAttenuation(0,m_nAttenuation);
```

void **SetClipMask** (VRECT vRect, Mask, long nRetVal)

Description

This method creates a custom area and mask for the video stream to be displayed at on a computer monitor. Use this command if you are not compiling with a Unicode option.

Parameters

vRect: A structure definition that stores border information (X Position, Y Position, length and width).

Mask: Used as argument for call to SafeArrayAccessData( ).

nRetVal: An integer variable that stores the result of the call. These results are the same as the values in the enumeration DEBIReturn. The enumeration and its defining values can be found in the "Definitions" section of this chapter, which starts on page 195.

Return Value

None

Example (VB)

N/A

Example (VC++)

```
ICVProServerPtr          m_pService;
VRECT                    Rect;
CClipMask                *pMask;
long                     nReturnVal;
m_pService->SetClipMask(Rect, pMask, nReturnVal);
```

VSReturn **SetClipMaskUni** (VRECT vRect)

Description

This method creates a custom area and mask for the video stream to be displayed at on a computer monitor. Use this command if you are compiling with a Unicode option.

Parameter

vRect: A structure definition that stores border information (X Position, Y Position, length and width).

Return Value

VSReturn: An enumeration value that indicates whether the execution of the command was successful. The definition of this enumeration can be found in the "Definitions" section of this chapter, which starts on page 195.

Example (VB)

N/A

Example (VC++)

```
ICVProServerPtr          m_pService;
VRECT                    m_pRect;
m_pService->SetClipMaskUni(m_pRect);
```

void **SetDestRect** (VRECT vRect, long nRetVal)

Description

This method sets the rectangular area of the monitor where the video stream output will be displayed. Use this command when you are compiling without the Unicode option set.

Parameters

vRect: A structure definition that stores border information (X position, Y position, length and width)

nRetVal: An integer value that stores the return value of the command. These results are the same as the values in the enumeration DEBIReturn. The enumeration and its defining values can be found in the "Definitions" section of this chapter, which starts on page 195.

Return Value

None

Example (VB)

N/A

Example (VC++)

```
ICVProServerPtr        m_pService;
VRECT                  pRect;
long                        m_nRetVal;
m_pService->SetDestRect(pRect, m_nRetVal);
```

VSReturn **SetDestRectUni** (VRECT vRect)

Description

This method sets the rectangular area of the monitor where the video stream output will be displayed. Use this command when you are compiling with the Unicode option set.

Parameter

vRect: A structure definition that stores border information (X position, Y position, length and width)

Return Value

VSReturn: An enumeration value that indicates whether the execution of the command was successful. The definition of this enumeration can be found in the "Definitions" section of this chapter, which starts on page 195.

Example (VB)

N/A

Example (VC++)

```
ICVProServerPtr          m_pService;
VRECT                    pRect;
long                         m_nRetVal;
m_pService->SetDestRectUni(pRect, m_nRetVal);
```

void **SetFilePos** (long HiOffset, long LowOffset, long *nRetVal)

Description

This method stops the decoder card from decoding, resets and starts decoding again from the file position indicated in its parameters.

Parameters

HiOffset: A value of type long that represents the first 32 bits of a 64-bit value indicating where along the MPEG-2 file the decoding is to start.

LowOffset: A value of type long that represents the next 32 bits of a 64-bit value indicating where along the MPEG-2 file the decoding to start.

nRetVal: A variable of type long that stores the result of the command. The values returned are the same as the values in the enumeration DEBIReturn. The enumeration and its defining values can be found in the “Definitions” section of this chapter, which starts on page 195.

Return Value

None

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lHigh as Long, lLow as Long, lResult as Long
' Set offset to 10000
lHigh = 0
lLow = 10000
.
' Call method to set file position
objCVProServer.SetFilePos(lHigh, lLow, lRetVal)
```

Example (VC++)

```
ICVProServerPtr          m_pService;
long                     High = (long)((__int64)tmpdbl >> 32);
long                     Low = (long)((__int64)tmpdbl & 0xffffffff);
m_pService->SetFilePos(&High, &Low, &m_nRetVal);
```

void **SetMasterReg** (long nReg, long dwValue, long bUpload, long nRetVal)

Description

This method sets the PCI register on the SAA7146 chip. This register is mainly used to set the brightness, contrast and saturation levels for the VGA output.

Parameters

nReg: A long value that indicates what command to associate with this register setting. For adjusting the colors, this value is usually 112.

dwValue: A long value that indicates the color settings that is to be set. Bits 16 to 23 of this value are used to indicate brightness. Bits 8 to 15 of this value are used to indicate contrast. Bits 0 to 7 of this value are used to indicate saturation.

bUpload: A long value that indicates whether or not to execute these new settings immediately. A value of 1 indicates to the decoder card to make the changes immediately and a value of 0 indicates to wait till the card is reset.

nRetVal: A variable of type long that stores the result of the execution of the command.

Return Value

None

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lColor as Long, lRetVal as Long
Dim     lBrightness as Long, lContrast as Long, lSaturation as Long
' Set brightness, contrast and saturation
lBrightness = 32
lContrast = 32
lSaturation = 50
' Shift values to the left to make one value
lColor = (lBrightness * (2 ^ 16)) Or (lContrast * (2 ^ 8)) Or (lSaturation)
' Call function
objCVProServer.SetMasterReg(112, lColor, 1, lRetVal)
```

Example (VC++)

```
#define VELA_BCS_CTRL 0x70
long color;
ICVProServerPtr                 m_pService;
m_pService->SetMasterReg(VELA_BCS_CTRL, color, TRUE, &m_nRetVal);
```

void **SetMidStreamStart** (BOOL newVal)

Description

This method sets a flag to indicate the API is in MidStreamStart mode. The SetMid-StreamStart mechanism is not currently used by the CineViewPro Client application.

Parameters

newVal: A Boolean value indicating whether to set or clear MidStreamStart mode.

Return Value

None

Example (VB)

```
Dim      objCVProServer as CineViewServer.CVProServer
' Clear MidStreamStart mode
objCVProServer.SetMidStreamStart(0)
```

Example (VC++)

```
ICVProServerPtr                m_pService;
M_Service->SetMidStreamStart(TRUE);
```

void **SetNextPlayListFile** (char * strFileName)

Description

This method is used to indicate the next MPEG-2 file to be played when the decoder is in back to back with no black frames in between mode. The programmer is to use this method when the application receives a windows message indicating it is ready to receive the next file.

Parameters

strFileName: A string that indicates the full path and filename of the MPEG-2 file that is to be played next when the current file is finished decoding.

Return Value

None

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     strFileName as String
' set file name of next MPEG file to play
strFileName = “D:\test1.mpg”
' Call function
objCVProServer.SetNextPlayListFile(strFileName)
```

Example (VC++)

```
ICVProServerPtr             m_pService;
TCHAR                       m_NextFilename[HX_N_MAX_FILE_NAME_CHARS];
m_pService->SetNextPlayListFile(m_NextFilename);
```

void **SetSrcRect** (VRECT vRect, long nRetVal)

Description

Sets the video or acquisition window rectangle.

Parameters

VRECT vRect: A structure containing a rectangle startX, StartY, width, and height parameters. The structure is defined in the “Definitions” section of this chapter, which starts on page 195.

nRetVal: A variable of type long that stores the result of the command. The values returned are the same as the values in the enumeration DEBIReturn. The enumeration and its defining values can be found in the “Definitions” section of this chapter.

Return Value

None

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     objRect as VRECT
Dim     lReturn as Long
' Set VRECT values in objRect then call function
objCVProServer.SetSrcRect(objRect, lReturn)
```

Example (VC++)

```
ICVProServerPtr          m_pService;
VRECT                    pVAcqRect
long                          nRetVal;
m_pService ->SetSrcRect(&pVAcqRect,&nRetVal);
```

VSReturn **SetSrcRectUni** (VRECT vRect)

Description

Unicode version of the method that sets the video or acquisition window rectangle.

Parameters

VRECT vRect: A structure containing a rectangle startX, StartY, width and height parameters. The structure is defined in the “Definitions” section of this chapter, which starts on page 195.

Return Value

VSReturn: An enumeration value that indicates whether the execution of the command was successful. The enumeration and its defining values can be found in the “Definitions” section of this chapter.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     objRect as VRECT
Dim     lReturn as Long
' Set values of VRECT in objRect and call function
lReturn = objCVProServer.SetSrcRectUni(objRect)
```

Example (VC++)

```
VSReturn pReturnVal
ICVProServerPtr          m_pService;
VRECT                    pVAcqRect;
m_pService ->SetSrcRect(&pVAcqRect, &pReturnVal);
```

long **SlowMotion** (long Speed)

Description

This method sets the decoder to display the same frame for a period of time to simulate slow motion.

Parameter

Speed: A value of type long that tells the decoder how many times each frame should be repeated to simulate slow motion. The values can range from 1 (very little slow motion) to 7 (very noticeable slow motion).

Return Value

The value returned is actually from the enumeration DEBIReturn. Its definition can be found in the back of this manual.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lRetVal as Long
' Start in slow motion at slowest possible speed.
lRetVal = objCVProServer.SlowMotion(7)
```

Example (VC++)

```
int                             m_nSlowSpeed;
ICVProServerPtr                 m_pService;
// Go in slow motion but at closest to normal speed
m_nSlowSpeed = 1;
m_nRetVal = m_pService->SlowMotion(m_nSlowSpeed);
```

VSReturn **StartStream** ()

Description

This method instructs the decoder card to begin feeding decoded video data into the DirectX video driver for display on a computer monitor.

Parameters

None

Return Value

VSReturn: An enumeration value that indicates whether the execution of the command was successful. The definition of this enumeration can be found in the "Definitions" section of this chapter, which starts on page 195.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     vsReturn as VSRetrun
vsReturn = objCVProServer.StartStream()
```

Example (VC++)

```
ICVProServerPtr                 m_pService;
VSReturn StatusReturned = m_pService->StartStream();
```

void **Stop** (long *nRetVal)

Description

This method instructs the decoder card to stop its current decode and place itself in a wait state.

Parameter

nRetVal: A variable of type long that stores the result of the command. The values returned are the same as the values in the enumeration DEBIReturn. The enumeration and its defining values can be found in the “Definitions” section of this chapter, which starts on page 195.

Return Value

None

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lRetVal as Long
objCVProServer.Stop(lRetVal)
```

Example (VC++)

```
ICVProServerPtr          m_pService;
long                         m_nRetVal;
m_pService->Stop(&m_nRetVal);
```

VSReturn **StopStream** ()

Description

This method stops the decoder card from feeding video data to the DirectX driver for display on a computer monitor.

Parameters

None

Return Value

VSReturn: An enumeration value that indicates whether the execution of the command was successful. The definition of this enumeration can be found in the "Definitions" section of this chapter, which starts on page 195.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     vsReturn as VSReturn
vsReturn = objCVProServer.StartStream()
.
.
.
vsReturn = objCVProServer.StopStream()
```

Example (VC++)

```
ICVProServerPtr             m_pService;
VSReturn                    m_nRetVal;
m_nRetVal = m_pService->StopStream();
```

long **StoreVideoSettings** (VelaVideoSettings Settings)

Description

This method saves decoder setting information into the EEPROM.

Parameters

LPVELAVIDEOSETTINGS: Settings structure containing settings information. See the “Definitions” section of this chapter, which starts on page 195.

Return Value

A long value that indicates success or failure. A value of zero indicates success where as any value other than zero indicates a problem.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     objSettings as VelaVideoSettings
Dim     lReturn as Long
' Set values of video in objSettings before making function call
lReturn = objCVProServer.StoreVideoSettings(objSettings)
```

Example (VC++)

```
ICVProServerPtr             m_pService;
VelaVideoSettings           pValue;
// Set values of video settings into pValue structure before calling function
HRESULT hr = m_pService->StoreVideoSettings(pValue);
```

CVProServer Class Property Descriptions

Long **AC3Mode**

Description

This property either gets or sets the value of whether AC-3 bit stream out is enabled or not. This also disables one audio channel.

Values can be as follows:

0 = AC-3 bit stream will be disabled

1 = AC-3 bit stream will be enabled

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lAC3Mode as Long
objCVProServer.AC3Mode = 1
' Card will output an AC-3 bit stream and disable an audio channel
lAC3Mode = ObjCVProServer. AC3Mode
' lAC-3Mode will contain the value 1
```

Example (C++)

```
ICVProServerPtr          m_pService;
BOOL                     bValue;
int                      m_nRetVal;
m_nRetVal = m_pService->put_ AC3Mode (bValue);
m_nRetVal = m_pService->get_ AC3Mode (&bValue);
```

Long **Audio20dbfs** (*Channel* As Integer)

Description

This property either gets or sets the PCM1716 DAC Mode Register to provide audio output level at one of two scales.

Parameter

Channel (Type: Integer): Indicates which channel number you want to get the value for or set the value to.

Values can be as follows:

1 = 24 dBm full scale
0 = 22 dBm full scale

Example (VB)

```
Dim      objCVProServer as CineViewServer.CVProServer
Dim      iChannel as Integer
iChannel = 0
' Sets audio of channel 0 to 24 dBm full scale
objCVProServer.Audio20dbfs(iChannel) = 1
```

Example (VC++)

```
ICVProServerPtr              m_pService;
int                          iChannel;
iChannel = 0;    // sets audio headroom of channel 0 to 20 db.
BOOL                         bMode;    // True for 20 db headroom, False for 18 db headroom
m_pService->put_Audio20dbfs(iChannel, bMode);
```

Integer **AudioPID** (*Channel* As Integer)

Description

This property either gets or sets the ID number of the audio stream in an encoded file. Values can range from 0 to 2^{32} –1, inclusive.

Parameter

Channel (Type: Integer): Indicates which channel number you want to get the value for or set the value to.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     iChannel as Integer
iChannel = 0
'Sets audio Program ID of channel 0 to 640
objCVProServer.AudioPID(iChannel) = 640
```

Example (VC++)

```
ICVProServerPtr             m_pService;
int                         iChannel;
int                         m n_AudioPID;
iChannel = 0;       // sets audio Program ID of ichannel to m_AudioPID
m_pService->put_AudioPID(iChannel, m_nAudioPID);
```

Long **AutoDetect**

Description

This property either gets or sets the CineView Pro decoder board to auto-detect the properties of an MPEG-2 file (Number of Lines, NTSC/PAL, 4:2:2 flag) via our own file parsing method or manually.

Values can be as follows:

0 = Settings are set manually by the user
1 = Settings are automatically detected using Vela's file parsing (Default)

Example (VB)

```
Dim      objCVProServer as CineViewServer.CVProServer
' decoder card will detect settings of MPEG-2 file beforehand
objCVProServer.AutoDetect = 0
```

Example (VC++)

```
ICVProServerPtr              m_pService;
BOOL                         newVal = true;
// decoder card will detect settings of MPEG-2 file beforehand
m_pService->put_AudioDetect(newVal);
```

Long **Bars**

Description

This property either gets or sets the output of composite color bars.

Values can be as follows:

0 = No color bars are generated to composite output
1 = Color bars are generated to composite output

Example (VB)

```
Dim      objCVProServer as CineViewServer.CVProServer
' bars are being generated and sent to composite output
ObjCVProServer.Bars = 1
```

Example (VC++)

```
ICVProServerPtr              m_pService;
BOOL                         m_bColorBar = true;
// generate color bars and send to composite output
m_nRetVal = m_pService->put_Bars(m_bColorBar);
```

Long **BlackLevel**

Description

This property either gets or sets the value of the level of black in the composite video output.

Values can range from 0 to 63, inclusive, where:

0 = Highest level of black available (maximum black)
63 = Lowest level of black available (minimum black)

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lLevel As Long
' Gets current Black level value
lLevel = objCVProServer.BlackLevel
' Sets Black level to minimum
objCVProServer.BlackLevel = 63
```

Example (VC++)

```
ICVProServerPtr          m_pService;
long                     lLevel;
long                          m_nBlackLevel = 63;
long                          m_nRetVal;
// sets current BlackLevel value
m_nRetVal = m_pService->put_BlackLevel(m_nBlackLevel);
// gets current BlackLevel value
m_nRetVal = m_pService->get_BlackLevel(&m_nBlackLevel);
```

Long **BlankLevel**

Description

This property either gets or sets the value of the level of blank (or white) in the composite video output.

Values can range from 0 to 63, inclusive, where:

0 = Highest level of blank (white) available (maximum blank or white)
63 = Lowest level of blank available (minimum white)

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lLevel As Long
 ' Gets current Blank (white) level value
lLevel = objCVProServer.BlankLevel
' Sets Blank level to minimum
objCVProServer.BlackLevel = 63
```

Example (VC++)

```
ICVProServerPtr          m_pService;
int                      m_nBlankLevel;
int                      m_nRetVal;
// gets current Blanking level value
m_nRetVal = m_pService->get_BlankLevel(&m_nBlankLevel);
// sets Blanking level to m_nBlankLevel value
m_nRetVal = m_pService->put_BlankLevel(m_nBlankLevel);
```

Long **BlankVideo**

Description

This property either gets or sets the option of whether or not video is to be output.

Values can be as follows:

0 = Video will be output normally
1 = Video will not be output at all

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lBlank As Long
 'lBlank stores value of whether or not composite video is blank or not
lBlank = objCVProServer.BlankVideo
'Blanking of video is turned off (decoded video is being sent to composite and
      'digital video output)
objCVProServer.BlankVideo = 0
```

Example (VC++)

```
ICVProServerPtr          m_pService;
long                          lBlank;
BOOL                          bBlank = true;
int                           m_nRetVal;
// blanks video from the decoder when bBlank is True // unblank when False
m_nRetVal = m_pService-->put_BlankVideo(bBlank);
```

Long **Brightness**

Description

This property either gets or sets the level of brightness (balance of light and dark shades) used in the VGA output (output placed on your computer monitor).

Values can range from 0 to 255, inclusive, where:

0 = No balance between light and dark shades used
255 = Highest level of balance available

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lBright as Long
'IBright stores level of brightness used in VGA output
lBright = objCVProServer.Brightness
'Level of brightness is set to 0 (no balance between light and dark shades)
objCVProServer.Brightness = 0
```

Example (VC++)

```
ICVProServerPtr          m_pService;
int                      m_nRetVal;
int                      newVal;    // set Brightness level to a new value
m_nRetVal = m_pService -> put_Brightness(newVal);
```

Long **BurstAmpitude**

Description

This property will either get or set the amplitude of the 3.579545 Mhz color reference subcarrier sample which is added to the back porch of the horizontal sync pulse.

Values can range from 0 to 127 where:

0 = No amplitude set
127 = Highest amplitude set

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lBurst As Long
objCVProServer.BurstAmplitude = 127
' Highest level it can be set to
lBurst = objCVProServer.BurstAmplitude
' lBurst is set to 127
```

Example (VC++)

```
ICVProServerPtr          m_pService;
int                      m_nBurstAmpl;    // set BurstAmplitude to a new value
m_nRetVal = m_pService->put_BurstAmplitude(m_nBurstAmpl);
// get current BurstAmplitude setting
m_nRetVal = m_pService->get_BurstAmplitude(&m_nBurstAmpl);
```

Long **ByteReorderMode**

Description

This property will either get or set the two bytes of information read from the closed caption information, which is transmitted during the vertical interval. Closed captioning commonly uses one of two modes. If the wrong mode is used during decoding, the caption information may appear garbled.

Values can be one of the following:

0 = Normal byte order for closed captioning (default)
1 = Swap byte order for closed captioning

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lByte As Long
objCVProServer.ByteReorderMode = 1
' Use swap byte method for closed captioning.
lByte = objCVProServer.ByteReorderMode
' lByte is set to 1
```

Example (VC++):

```
ICVProServerPtr             m_pService;
BOOL                        m_bByteSwap = true;
// sets decoder mode to swap closed caption bytes
m_pService->put_ByteReorderMode(m_bByteSwap);
```

Long **ChromaPhase**

Description

This property will either get or set a phase offset applied to both the color burst and the phase-modulated hue and saturation information, or chroma, which is added to each line of video.

Values can range from 0 to 255, inclusive, where:

0 = No offset used for each video line
255 = Maximum offset used for each video line

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lChroma As Long

objCVProServer.ChromaPhase = 127
' middle of the road phase offset
lChroma = objCVProServer.ChromaType
' lChroma is now set to 127
```

Example (VC++)

```
ICVProServerPtr          m_pService;
int                      m_nChroma;
// sets decoder chroma phase to m_nChroma value
m_nRetVal = m_pService->put_ChromaPhase(m_nChroma);
```

Long **ChromaType**

Description

This property either gets or sets the value that indicates the chroma type the card is set to use for decoding of MPEG files.

Values can be as follows:

0 = 4:2:0 chroma type
1 = 4:2:2 chroma type
2 = 4:4:4 chroma type

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lChroma As Long
lChroma = objCVProServer.ChromaType
objCVProServer.ChromaType = 1 '4:2:2 chroma
```

Example (VC++)

```
ICVProServerPtr          m_pService;
BOOL                     m_b422 = true;
m_nRetVal = m_pService->put_ChromaType(m_b422);
```

Long **Contrast**

Description

This property either gets or sets the contrast value (range between lightest tones and darkest tones) used in the VGA output.

Values can range from 0 to 100, inclusive, where:

0 = No contrast
100 = Highest level of contrast available

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lContrast as Long
lContrast = objCVProServer.Contrast
objCVProServer.Contrast = 100 'maximum contrast level set
```

Example (VC++)

```
ICVProServerPtr          m_pService;
int                      nValue;
int                      m_nRetVal;
m_nRetVal = m_pService->put_Contrast(nValue);
```

Long **FirstVideoLine**

Description

This property either gets or sets the setting of the first active line of video, which can either be line 6 or line 7.

Values can be as follows:

6 = First active line of video is line 6 (default)
7 = First active line of video is line 7

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lFirst as Long
lFirst = objCVProServer.FirstVideoLine
objCVProServer.FirstVideoLine = lFirst
```

Example (VC++)

```
ICVProServerPtr          m_pService;
int                      m_nRetVal;
int                      nValue;
m_nRetVal = m_pService->put_FirstVideoLine(nValue);
```

Long **FreezeFieldMode**

Description

This property either gets or sets the decoder pause mode where a single field is displayed as a frame, instead of alternating between two fields. When the decoder is paused, FreezeFieldMode has the effect of freezing apparent motion or jitter between two fields.

Values can be as follows:

0 = Freeze on field as if it is a frame
1 = Freeze on frame (default)

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lFreeze as Long
objCVProServer.FreezeFieldMode = 1
' Freeze on Frame when Pause command is executed.
lFreeze = objCVProServer.FirstVideoLine
' lFreeze now is set to 1.
```

Example (VC++)

```
ICVProServerPtr          m_pService;
int                      m_nRetVal;
BOOL                     m_bFrzFrame;
// freeze on field when Pause command is executed
m_nRetVal = m_pService->put_FreezeFieldMode(m_bFrzFrame);
```

Long **GainU**

Description

This property either gets or sets the amplitude of the (B' - Y) or Blue minus luminance color difference signal which is added to each line of video.

Values can range from 0 to 255, inclusive, where:

0 = No amplitude of Blue minus luminance color
255 = Maximum amplitude of Blue minus luminance color

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lGainU as Long
objCVProServer.GainU = 255
' Maximum amplitude of Blue minus luminance color set.
lGainU = objCVProServer.GainU
' lGainU is now set to 255.
```

Example (VC++)

```
ICVProServerPtr          m_pService;
int                      m_nGainU;
int                      m_nRetVal;
m_nRetVal = m_pService->put_GainU(m_nGainU);
m_nRetVal = m_pService->get_GainU(&m_nGainU);
```

Long **GainV**

Description

This property either gets or sets the amplitude of the (R' - Y) or Red minus luminance color difference signal which added to each line of video.

Values can range from 0 to 255, inclusive, where:

0 = No amplitude of Red minus luminance color
255 = Maximum amplitude of Red minus luminance color

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lGainV as Long
objCVProServer.GainV = 255
' Maximum amplitude of Red minus luminance color set.
lGainV = objCVProServer.GainV
' lGainV is now set to 255.
```

Example (VC++)

```
ICVProServerPtr          m_pService;
int                      m_nGainV;
int                      m_nRetVal;
m_nRetVal = m_pService->put_GainV(m_nGainV);
m_nRetVal = m_pService->get_GainV(&m_nGainV);
```

Long **GenLock**

Description

This property either gets or sets the value on whether the decoder hardware will lock on frequency and phase using an external source or not.

Value can be as follows:

0 = CineView Pro does not use outside source for locking (standalone mode) (default)

1 = CineView Pro uses an external source for frequency and phase locking

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lLock as Long
lLock = objCVProServer.GenLock
objCVProServer.GenLock = 1
'Use outside source for locking…source is assumed to be plugged in
```

Example (VC++)

```
ICVProServerPtr          m_pService;
BOOL                     m_bGenlock;
int                      m_nRetVal;
m_nRetVal = m_pService->put_GenLock(m_bGenlock);
```

Long **HorizontalPhase**

Description

This property either gets or sets the delay between the decoder output horizontal sync and the input genlock sync source sync.

Values are dependent on what format the MPEG file was encoded in. For the NTSC format, values can range from 0 to 1715, inclusive. For the PAL format, values can range from 0 to 1727, inclusive. A value of zero indicates no delay where as the maximum value indicates the maximum amount of delay to set.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lPhase as Long
objCVProServer.HorizontalPhase = 1715
' Set maximum delay for NTSC
lPhase = objCVProServer.HorizontalPhase
' lPhase is set to 1715
```

Example (VC++)

```
ICVProServerPtr          m_pService;
int                      m_nRetVal;
int                      m_nHorizontalPhase;
m_nRetVal = m_pService->put_HorizontalPhase(m_nHorizontalPhase);
m_nRetVal = m_pService->get_HorizontalPhase(&m_nHorizontalPhase);
```

Long **MaxLineResolution**

Description

This property either gets or sets the number of lines, maximum, that will be displayed, in output. This setting is dependent on whether the file to be decoded, is in either NTSC or PAL format.

Values can be as follows:

0 = Lower number of maximum lines (for NTSC is 480, PAL is 576)
1 = Higher number of maximum lines (for NTSC is 512, PAL is 608)

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lMax as Long
objCVProServer.NTSC = 1 'NTSC format is expected of the encoded file.
objCVProServer.MaxLineResolution = 1
'Sets card to expect 512 vertical lines (set for NTSC)
lMax = objCVProServer.MaxLineResolution 'lMax is set to 1.
```

Example (VC++)

```
ICVProServerPtr          m_pService;
int                      m_nRetVal;
BOOL                     m_bFileHiResolution;
m_nRetVal = m_pService->put_MaxLineResolution(m_bFileHiResolution);
m_nRetVal = m_pService->get_MaxLineResolution(&m_bFileHiResolution);
```

Long **MuxedStreamType**

Description

This property either gets or sets the type of stream that is being used in decoding.

Value can be as follows:

0 = Not an MPEG Stream
1 = A System Stream
2 = A Program Stream
3 = A Transport Stream
4 = A Video Elementary Stream
5 = An Audio Elementary Stream
6 = A Video Packetized Elementary Stream
7 = An Audio Packetized Elementary Stream

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lMuxed as Long
lMuxed = objCVProServer.MuxedStreamType
objCVProServer.MuxedStreamType = 2 'Sets decoder card to decode a Program Stream.
```

Example (VC++)

```
ICVProServerPtr          m_pService;
int                      m_nRetVal;
int                      nValue;
m_nRetVal = m_pService->put_MuxedStreamType(nValue);
m_nRetVal = m_pService->get_MuxedStreamType(&nValue);
```

Long **NTSC**

Description

This property either gets or sets the value on whether the MPEG file to be decoded is in NTSC or PAL format.

Value can be as follows:

0 = PAL format
1 = NTSC format

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lType as Long
objCVProServer.NTSC = 1 ' Sets decoder card to expect an NTSC file
lType = objCVProServer.NTSC 'lType is set to 1.
```

Example (VC++)

```
ICVProServerPtr          m_pService;
BOOL                     m_bNTSC;
int                      m_nRetVal;
m_nRetVal = m_pService->put_NTSC(m_bNTSC);
m_nRetVal = m_pService->get_NTSC(&m_bNTSC);
```

Long **NumDecoders**

Description

This property is a read-only property that returns the number of CineView decoder cards it finds installed on your computer. The maximum number of cards that can be installed in a PC is 4.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lType as Long
lCount = objCVProServer.NumDecoders
'lCount has the number of decoders found installed in the computer
```

Example (VC++)

```
ICVProServerPtr          m_pService;
short                    *p_Val;
int                      m_nRetVal;
m_nRetVal = m_pService->get_NumDecoders(&pVal);
```

Long **PauseOnFFMode**

Description

This property either gets or sets the value of whether the feature to turn on pause when the first frame is decoded is enabled or not.

Value can be as follows:

0 = Pause on First Frame is Disabled (default)
1 = Pause on First Frame is Enabled

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lPause as Long
objCVProServer.PauseOnFFMode = 1
' Card will pause on first frame of decoded file when play begins
lPause = objCVProServer.PauseOnFFMode
' lPause will contain the value 1.
```

Example (VC++)

```
ICVProServerPtr          m_pService;
BOOL                     bValue;
int                      m_nRetVal;
m_nRetVal = m_pService->put_PauseOnFFMode(bValue);
m_nRetVal = m_pService->get_PauseOnFFMode(&bValue);
```

Long **PlayListMode**

Description

This property either gets or sets the value of whether to enable or disable back to back playback with no black frames in between.

Value can be as follows:

0 = BackToBack playback with no black is disabled (default)
1 = BackToBack playback with no black is enabled

Example (VB)

```
Dim      objCVProServer as CineViewServer.CVProServer
objCVProServer.PlayListMode = 1
' Back to Back is enabled.
' Code to demonstrate how to use the back to back with no black feature is available in the
' Visual Basic sample code of the CineView Pro. Make arrangements with your sales
' representative to obtain the password to install this code.
```

Example (VC++)

```
ICVProServerPtr                m_pService;
// back to back is enabled.
// Code used to demonstrate how to use the back to back with no black feature is available in the
// Visual C++ sample code of the CineView Pro. Make arrangements with your sales
// representative to obtain the password to install the code.
// place API in playlist mode
m_pService->PutPlayListMode(true);
// take API out of playlist mode
m_pService->PutPlayListMode(false);
```

Long **ProgramID**

Description

This property either gets or sets the Identification number (ID) of a multiplexed MPEG-2 stream (has both audio and video data). Since changing the ID of a stream can affect it for future play, please use this property with caution.

Values can range from 0 to 32678, inclusive

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lProgID as Long
lProgID = objCVProServer.ProgramID
' lProgID now stores the ID number of the stream. This ID was set during the encode session
' that created the file. For MPEG-2 files created by Vela's Argus system, the program ID defaults
' to 1.
objCVProServer.ProgramID = 15
' sets program ID of encoded file to 15 for this session of decoding only (not recommended).
```

Example (VC++)

```
ICVProServerPtr          m_pService;
int                      nPIDValue;
int                      nChannel;
m_pService->put_ProgramID(nChannel, nPIDValue);
```

Long **Saturation**

Description

This property either gets or sets the VGA saturation level. Saturation corresponds to the amount of color present. A hue at maximum saturation appears as a pure color with no white added, while a less saturated hue appears more pastel.

Values can range from 0 to 100, inclusive, where:

0 = No saturation allowed at all
100 = Maximum level of saturation

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lSaturation as Long
objCVProServer.Saturation = 100
' Maximum level of saturation set for VGA display
lSaturation = objCVProServer.Saturation
' lSaturation is now set to 100
```

Example (VC++)

```
ICVProServerPtr          m_pService;
int                          n_RetValue;
int                          nValue;
m_pService ->put_Saturation(nValue);
n_RetValue = m_pService ->gett_Saturation(&nValue);
```

Long **ShowVGADisplay**

Description

This property gets or sets the value of whether to show the updated VGA display.

Values can be as follows:

0 = VGA will not be showing

1 = VGA will be showing

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lVGAShowing as Long
objCVProServer.ShowVGADisplay = 1
' Server will show the updated VGA
lVGAShowing = ObjCVProServer.ShowVGADisplay
' lVGAShowing will contain the value 1
```

Example (C++)

```
ICVProServerPtr             m_pService;
BOOL                        bValue;
int                         m_nRetVal;
m_nRetVal = m_pService->put_ShowVGADisplay(bValue);
m_nRetVal = m_pService->get_ShowVGADisplay(&bValue);
```

Long **SlowMotionRate**

Description

This property either gets or sets the rate of motion the decoder will display in output when it is in slow motion mode. Slow motion is accomplished by repeating frames. The slow motion value refers to the number of times each frame is to be repeated.

Values can range from 0 to 7, inclusive where:

0 = video is displayed at normal speed
7 = video is displayed at slowest possible rate

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lMotionRate as Long
objCVProServer.SlowMotionRate = 7
objCVProServer.SlowMotion(objCVProServer.SlowMotionRate)
' Card executes in slow motion at the slowest possible rate
lMotionRate = objCVProServer.SlowMotionRate
' lMotionRate now stores the value of 7
.
.
.
objCVProServer.Resume
' card resumes play in normal mode.
```

Example (VC++)

```
ICVProServerPtr          m_pService;
int                      nValue;
int                      m_nRetValue;    // update decoder slow motion rate
m_nRetValue = m_pService -->put_SlowMotionRate(nValue);
// get current decoder slow motion rate
m_nRetValue = m_pService -->get_SlowMotionRate(nValue);
```

Long **VGADisplay**

Description

This property either gets or sets the value of whether to use the updated VGA display or the old VGA.

Value can be as follows:

0 = Old VGA will be used

1 = Updated VGA will be used

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lVGADisplay as Long
objCVProServer.VGADisplay = 1
' Server will use the updated VGA
lVGADisplay = ObjCVProServer.VGADisplay
' lVGADisplay will contain the value 1
```

Example (C++)

```
ICVProServerPtr             m_pService;
BOOL                        bValue;
int                         m_nRetVal;
m_nRetVal = m_pService->put_VGADisplay(bValue);
m_nRetVal = m_pService->get_VGADisplay(&bValue);
```

Long **VideoPID**

Description

This property either gets or sets the Identification number (ID) of the video part of a multiplexed MPEG-2 stream. Since changing the ID of a stream can affect it for future play, please use this property with caution.

Values can range from 0 to 232 -1, inclusive.

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lProgID as Long
lProgID = objCVProServer.VideoPID
' lProgID now stores the ID number of the video part of the stream. This ID was set during the
' encode session that created the file. For MPEG-2 files created by Vela's Argus system, the
' video program ID defaults to 512.
objCVProServer.VideoPID = 15
' sets the video program ID of encoded file to 15 for this session of decoding only
    ' (not recommended).
```

Example (VC++)

```
ICVProServerPtr             m_pService;
int                             nValue;
m_nRetVal = m_pService->put_VideoPID(nValue);
m_nRetVal = m_pService->get_VideoPID(&nValue);
```

Long **VideoReference**

Description

This property either gets or sets the amplitude of the composite video output, which includes video plus sync.

Values can range from 0 to 100, inclusive, where:

0 = No amplitude of the composite video output
100 = Maximum amplitude of the composite video output

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lVideo as Long
objCVProServer.VideoReference = 100
' Maximum amplitude of composite video output set.
lVideo = objCVProServer.VideoReference
' lVideo now stores the value 100.
```

Example (VC++)

```
ICVProServerPtr          m_pService;
int                      nValue;
m_nRetVal = m_pService->put_VideoReference(nValue);
m_nRetVal = m_pService->get_VideoReference(&nValue);
```

Long **ZeroIRE**

Description

This property either gets or sets the black level to either 0 IRE units or 7.5 IRE units and then reinitializes the 7185 digital video encoder hardware.

Value can be as follows:

0 = 7.5 IRE units
1 = 0 IRE units

Example (VB)

```
Dim     objCVProServer as CineViewServer.CVProServer
Dim     lZeroIRE as Long
objCVProServer.ZeroIRE = 0
' card is set to use 7.5 IRE units in black level
lZeroIRE = objCVProServer.ZeroIRE
' lZeroIRE is set to a value of 0
```

Example (VC++)

```
ICVProServerPtr                 m_pService;
BOOL                                    m_b0IRE;
m_nRetVal = m_pService->put_ZeroIRE(m_b0IRE);
m_nRetVal = m_pService->get_ZeroIRE(&m_b0IRE);
```

CVProServer Class Event Descriptions

There are no Events to describe.

OnScreenDisplay Class Method Descriptions

long **EnableOSD** (bOn As BOOL, bBlendOn As BOOL, nRegionNum As Long)

Description

Enables or disables the OSD region specified. This command can be executed while a clip is playing in order to disable or enable the currently activated OSD region. You are not allowed to change the bBlendOn value of a region once the clip has started playing.

To display multiple pictures you must first load all of the bitmaps and then cyclically call EnableOSD with bOn equal to 1 and a value for nRegionNum equal to the bitmap you want displayed. There is no need to disable the previous OSD region before displaying the next.

Only one OSD region can be enabled at any time.

Parameters

BOOL bOn: Value to enable or disable OSD

0 = Disable OSD region
1 = Enable OSD region

BOOL bBlendOn: Value to enable or disable blending between the video and OSD region.

0 = Disable blending for OSD region
1 = Enable blending for OSD region

long nRegionNum: Specifies which one of the loaded OSD regions is to receive this command. If only one bitmap is loaded this value must be set to 0.

Range: 0 to 15

Return Value

Is not implemented.

Example (VB)

```
Dim     objCVProOSD as CineViewServerLib.OnScreenDisplay
Dim     lTurnOSDOn
Dim     lBlend
lTurnOSDOn = 1
lBlend = 1
objCVProOSD.EnableOSD lTurnOSDOn, lBlend, 0
```

Example (VC++)

```
IOnScreenDisplayPtr     m_pOSD;
BOOL                    m_bEnableBlending = true;
m_pOSD->EnableOSD(true, m_bEnableBlending, 0);
```

void **Initialize** (Card As Integer, bReserveMemory As BOOL)

Description

Initializes current decoder to receive OSD commands.

Parameters

short Card: Specifies which decoder board is to be initialized.

Range: 0 to 3

BOOL bReserveMemory: specifies whether or not memory is to be reserved for the OSD region.

0 = Do not reserve OSD memory
1 = Reserve OSD memory

Return Value

void

Example (VB)

```
Dim     objCVProOSD as CineViewServerLib.OnScreenDisplay
Dim     lCardNo
lCardNo = 0
ObjCVProOSD.Initialize lCardNo, 1
```

Example (VC++)

```
IOnScreenDisplayPtr          m_pOSD;
m_pOSD->Initialize(0, true);
```

long **LoadBitmap** (bstrFileName As BSTR, x As Long, y As Long, nRegionNum As Long, bTransBKGnd As BOOL, lBlendFactor As Long)

Description

Loads the bitmap file to be used for OSD. To load multiple bitmaps each bitmap must be given a unique nRegionNumber. Up to 16 bitmaps can be loaded at one time. The X and Y coordinates specify upper left corner of the bitmap.

The width of the bitmap must be divisible by 8.

Parameters

BSTR bstrFileName: Contains the path and filename of the bitmap that is to be loaded.

long x: Horizontal coordinate for placement of the OSD region.
Range: 0 to 700

long y: Vertical coordinate for placement of the OSD region.
Range: 0 to 240 for NTSC clips; 0 to 304 for PAL clips

long nRegionNum: Identifies which region number is being loaded.
Range: 0 to 15

BOOL bTransBKGnd: Value used to indicate if the OSD background should be cleared.

0 = Do not clear the OSD background
1 = Clear the OSD background

long lBlendFactor: Value indicating the level of blending between the OSD region and the video.

100 = No video shows through the OSD region

75 = 75% of the OSD region is visible; 25% of the video shows through the bitmap

50 = 50% of the OSD region is visible; 50% of the video shows through the bitmap

25 = 25% of the OSD region is visible; 75% of the video shows through the bitmap

Return Value

0 = Loaded successfully
3 = The path could not be found
4 = The file was not found
5 = Access to the file has been denied
12 = The width of the bitmap file is not divisible by 8

Example (VB)

```
Dim     objCVProOSD as CineViewServerLib.OnScreenDisplay
lReturn = MyEvents.OSDObject.LoadBitmap(txtBMPFile.Text, lOSDXPos, lOSDYPos, 0, _
        bOSDClearBKGrnd, lOSDBlendFactor)
if lReturn <> 0 then
        'An error has occurred while trying to load the bitmap
End if
```

Example (VC++)

```
IOnScreenDisplayPtr     m_pOSD;
int                     m_Xpos = 50;
int                     m_Ypos = 50;
int                     m_nRegionCount = 0;
bool                    m_bClearBackground = true;
long                        m_lBlendLevel = 75;
int                     errVal = m_pOSD->LoadBitmap("C:\\temp.bmp",
                            m_XPos,m_YPos, m_nRegionCount,
                            m_bClearBackground, m_lBlendLevel);
if (errVal != 0)
{
        //There has been a problem loading the bitmap
}
```

long **LoadRegion** (bstrFileName As String, x As Long, y As Long, nRegionNum As Long)

Description

Used to activate a specific bitmap in a multiple bitmap OSD animation.

This method has not been implemented.

Instead use **EnableOSD** with nRegionNum set to the region number of the Bitmap you want to load.

Parameters

Not implemented

Return Value

Not implemented

Example (VB)

N/A

Example (VC++)

N/A

OnScreenDisplay Class Property Descriptions

There are no Properties to describe.

OnScreenDisplay Class Event Descriptions

There are no Events to describe.

FrameControlledPlayback Class Method Descriptions

int **nRetVal FrameAccuratePlay** (BSTR bstrFileName, int StreamType)

Description

This method is used to start a frame-accurate play. This is used instead of the Play method in the CVProServer interface. You must set frame-accurate properties before calling this method.

Parameter

bstrFileName: A string value that indicates the name of the MPEG file that is going to be used in the FrameControlledPlayback interface.

StreamType: An integer value that indicates the type of stream the MPEG file is. The values can be as follows:

1 = Video Elementary

3 = Video Packetized Elementary stream

4 = Multiplexed stream

Return Value

FrameAccuratePlay returns a value less than 0 if an error occurs. If the play is successful, 0 is returned.

Example (VB)

N/A

Example (VC++)

```
IFrameControlledPlaybackPtr     pFrameControlledPlayback;
TCHAR                           tempFilename[200];

Wcscpy(tempFilename, m_csFilename);
.
.
.
pFrameControlledPlayback->FrameAccuratePlay(tempFilename, m_nFileType);
```

Void **Initialize** (short nCardNum)

Description

Use Initialize to prepare a frame-accurate interface for playing back a file.

Parameters

nCardNum: Indicates the card number that you want the current instance of the Frame-ControlledPlayback interface to use. nCardNum may range from 0 to 3, inclusive.

Return Value

This method does not return a value.

Example (VB)

N/A

Example (VC++)

```
IFrameControlledPlaybackPtr     pFrameControlledPlayback;
short                           nCardNum;
nCardNum                        = 0;

pFrameControlledPlayback->Initialize(nCardNum);
.
.
.
pFrameControlledPlayback->FrameAccuratePlay(tempFilename, m_nFileType);
```

FrameControlledPlayback Class Property Descriptions

short **EndControlType**

Description

This property allows you to specify which property the FrameControlledPlayback interface will use to end the playback.

Parameters

A short specifying which method will be used to end playback.
Values can be as follows:

0 = FrameEnd. Specifies that the EndFrame property will be used to end playback.

1 = TimeCodeEnd. Specifies that the EndTime property will be used to end playback.

2 = FramesPlay. Specifies that the FrameDuration property will be used to end playback.

Example (VB)

N/A

Example (VC++)

```
IFrameControlledPlaybackPtr     pFrameControlledPlayback;
long                            nFrameDuration;
nFrameDuration                  = 100;

pFrameControlledPlayback->put_FrameDuration(nFrameDuration);
.
.
.
pFrameControlledPlayback->put_EndControlType(2);
.
.
.
pFrameControlledPlayback->FrameAccuratePlay(tempFilename, m_nFileType);
```

long **EndFrame**

Description

This property sets how many frames into the file you wish to stop playing. You must also specify a value of FrameEnd in the EndControlType property in order to end playback at the given frame.

Parameter

This value must be set to a value higher than the starting frame and must be a valid frame number.

Example (VB)

N/A

Example (VC++)

```
IFrameControlledPlaybackPtr     pFrameControlledPlayback;
long                            nEndFrame;
nEndFrame                       = 100;

pFrameControlledPlayback->put_EndFrame(nEndFrame);
.
.
.
pFrameControlledPlayback->put_EndControlType(0);
.
.
.
pFrameControlledPlayback->FrameAccuratePlay(tempFilename, m_nFileType);
```

BSTR **EndTime**

Description

This property is used to set the time code at which you wish to stop playing. You must also specify a value of TimeCodeEnd in the EndControlType property in order to end playback at the given time code.

Parameter

EndTime must have a time code value that is greater than the specified start point. It must also be a valid time code in the file you are trying to play.
EndTime must have the following format:

hh:mm:ss:ff

hh= hours

mm= minutes

ss = seconds

ff = frames

Example (VB)

N/A

Example (VC++)

```
IFrameControlledPlaybackPtr     pFrameControlledPlayback;
CString                         csEndTime;
csEndTime                       = _T("00:26:15:08");

BSTR time = _bstr_t(csEndTime)
pFrameControlledPlayback->put_EndTime(time);
.
.
.
pFrameControlledPlayback->put_EndControlType(1);
.
.
.
pFrameControlledPlayback->FrameAccuratePlay(tempFilename, m_nFileType);
```

BOOL **FrameControlMode**

Description

Not Implemented.

long **FrameDuration**

Description

This property specifies the number of frames from the start point at which the playback will stop. You must also specify a value of FramesPlay in the EndControlType property in order to end playback at the given duration.

Parameters

Any valid number of type long that does not cause the duration plus starting point to go past the end of the file.

Example (VB)

N/A

Example (VC++)

```
IFrameControlledPlaybackPtr     pFrameControlledPlayback;
long                            nFrameDuration;
nFrameDuration                  = 100;

pFrameControlledPlayback->put_FrameDuration(nFrameDuration);
.
.
.
pFrameControlledPlayback->put_EndControlType(2);
.
.
.
pFrameControlledPlayback->FrameAccuratePlay(tempFilename, m_nFileType);
```

BOOL **PauseOnFF**

Description

Set this property to True to enable pause on the first frame. The playback will pause on the specified starting point and another play command must be issued in order to resume play.

If the starting point given happens to be an I-frame in the given file, the FrameControlledPlayback interface will set the CVProServer interface PauseOnFFMode property to True in order to accurately pause on the first frame.

If your next playback is not done with the FrameControlledPlayback interface's FrameAccuratePlay method and you do not want to pause on the first frame it may be necessary to set the PauseOnFFMode property to False before you begin playing.

Parameters

A BOOL specifying pause on first frame, enabled or disabled.
Values can be as follows:

1 = Pause on first frame is enabled.

0 = Pause on first frame is disabled (default).

Example (VB)

N/A

Example (VC++)

```
IFrameControlledPlaybackPtr     pFrameControlledPlayback;
ICVProServerPtr                 pService;
.
.
.
pFrameControlledPlayback->put_PauseOnFF(1);
.
.
.
pFrameControlledPlayback->FrameAccuratePlay(tempFilename, m_nFileType);
.
.
.
//Be sure to clear FF mode in CVProServer prior to doing normal playback
pService->put_PauseOnFFMode(0);
pService->Play(tempFilename, m_nFileType);
```

BOOL **PauseOnLF**

Description

Set this property to True to enable pausing on the last frame. The playback will pause on the specified end point. This property must be set prior to playback.

Parameters

A BOOL specifying pause on last frame, enabled or disabled.
Values can be as follows:

1 = Pause on last frame is enabled.

0 = Pause on last frame is disabled (default).

Example (VB)

N/A

Example (VC++)

```
IFrameControlledPlaybackPtr     pFrameControlledPlayback;
ICVProServerPtr                 pService;
.
.
.
pFrameControlledPlayback->put_PauseOnFF(1);
pFrameControlledPlayback->FrameAccuratePlay(tempFilename, m_nFileType);
```

long **PreBlack**

Description

Not implemented.

short **StartControlType**

Description

This property allows you to specify which property the FrameControlledPlayback interface will use to start the playback.

Parameters

A short specifying which method will be used to start playback.
Values can be as follows:

0 = FrameStart. Specifies that the StartFrame property will be used to start playback.

1 = TimeCodeStart. Specifies that the StartTime property will be used to start playback.

Example (VB)

N/A

Example (VC++)

```
IFrameControlledPlaybackPtr     pFrameControlledPlayback;
long                            nStartFrame;
nStartFrame                     = 10;

pFrameControlledPlayback->put_StartFrame(nStartFrame);
.
.
.
pFrameControlledPlayback->put_StartControlType(0);
.
.
.
pFrameControlledPlayback->FrameAccuratePlay(tempFilename, m_nFileType);
```

long **StartFrame**

Description

This property sets how many frames into the file you wish to start playing. You must also specify a value of FrameStart in the StartControlType property in order to start playback at the given frame.

Parameter

This value must be set to a value less than the ending frame and must be a valid frame number.

Example (VB)

N/A

Example (VC++)

```
IFrameControlledPlaybackPtr     pFrameControlledPlayback;
long                            nStartFrame;
nStartFrame = 10;

pFrameControlledPlayback->put_StartFrame(nStartFrame);
.
.
.
pFrameControlledPlayback->put_StartControlType(0);
.
.
.
pFrameControlledPlayback->FrameAccuratePlay(tempFilename, m_nFileType);
```

BSTR **StartTime**

Description

This property is used to set the time code at which you wish to begin playing. You must also specify a value of TimeCodeStart in the StartControlType property in order to start playback at the given time code.

Parameter

StartTime must have a time code value that is less than the specified end point. It must also be a valid time code in the file you are trying to play.
StartTime must have the following format:

hh:mm:ss:ff

hh= hours

mm= minutes

ss = seconds

ff = frames

Example (VB)

N/A

Example (VC++)

```
IFrameControlledPlaybackPtr     pFrameControlledPlayback;
CString                         csStartTime;
csStartTime                     = _T("00:23:45:00");

BSTR time = _bstr_t(csStartTime)
pFrameControlledPlayback->put_StartTime(time);
.
.
.
pFrameControlledPlayback->put_StartControlType(1);
.
.
.
pFrameControlledPlayback->FrameAccuratePlay(tempFilename, m_nFileType);
```

Definitions

```
typedef enum                    // VelaPCI return codes
    {
        DEBI_NO_ERROR = 0,              // No error
        CONFIGURE_DENC_ERROR,           // error configuring DENC 7185
        CONFIGURE_DMSD_ERROR,           // error configuring DMSD 7111
        DEBI_FILE_OPEN_ERROR,           // error opening file
        DEBI_ISR_ERROR,                 // error registering interrupt service routine
        DEBI_DMA_TIMEOUT_ERROR,         // error sending data to decoder
        DEBI_NULL_POINTER,              // null pointer passed to function
        DEBI_MEMORY_ERROR,              // decoder did not accept command
        DEBI_LOWER_VOLUME_LIMIT,        // lower audio volume limit reached
        DEBI_UPPER_VOLUME_LIMIT,        // upper audio volume limit reached
        DEBI_TIMEOUT_ERROR,             // transfer timed out
        DEBI_ERROR,                     // General error encountered
        DSP_COMMAND_ERROR,              //
        DEBI_OUT_OF_RANGE,              // Value out of Range
        DEBI_COMMAND_ERROR,             // invalid command for current decoder
                                            state
        DSP_COMPLETION_ERROR,           // completion code from DSP not found
        DEBI_MICROCODE_FILE_ERROR,// open of DSP microcode file failed.
        DSP_WRAP_DATA_ERROR,            // mailbox wrap around test returned bad
                                            data
        DSP_WRAP_TIMEOUT_ERROR,         // mailbox wrap around test timed out
        DEBI_TIMECODE_ERROR,            // error reading SMPTE time code
        DEBI_STARTATFRM_ERROR
    } DEBIReturn;

typedef /* [uuid] */ struct DefaultCVProSettings
    {
    int HPhase;
    int CPhase;
    int BlackLevel;
    int BlankLevel;
    int UGain;
    int VGain;
```

```
    int BurstAmplitude;
    int Brightness;
    int Contrast;
    int Saturation;
    }  DefaultCVProSettings;

typedef struct
DefaultCVProSettings __RPC_FAR *LPDEFAULTCVPROSETTINGS;

typedef /* [uuid] */ struct VelaStreamInfo
    {
    VelaStreamType StreamType;
    int HorizRes;
    int VertRes;
    double FrameRate;
    int VideoBitRate;
    int AudioBitRate;
    MuxedStreamType MuxedStreamType;
    long MuxedRate;
    int IsInterlaced;
    VelaChromaFormat ChromaFormat;
    int numProgramEntries;
    VelaProgramInfo VelaProgramInfo[ 60 ];
    int numVideoStreams;
    VelaVideoStreamInfo VelaVideoInfo[ 30 ];
    int numAudioStreams;
    VelaAudioStreamInfo VelaAudioInfo[ 30 ];
    }  VelaStreamInfo;

typedef struct VelaStreamInfo __RPC_FAR *LPVELASTREAMINFO;

typedef /* [uuid] */ struct VelaVideoSettings
    {
    int nHorizPhase;
    int nChromaPhase;
    int nBlackLevel;
    int nBlankLevel;
    int nGainU;
```

```
    int nGainV;
    int nBurstAmplitude;
    int nVideoRef;
    int nFirstLine;
    }   VelaVideoSettings;

typedef struct VelaVideoSettings __RPC_FAR *LPVELAVIDEOSETTINGS;

typedef /* [uuid] */ struct VRECT
    {
    long nStartX;
    long nStartY;
    long nWidth;
    long nHeight;
    }   VRECT;

typedef struct VRECT __RPC_FAR *LPVRECT;

enum VSReturn
    {   VS_ERR_NONE                 = 0,
        VS_ERR_OPEN_CONFIG          = VS_ERR_NONE + 1,
        VS_ERR_OPEN_INUSE           = VS_ERR_OPEN_CONFIG + 1,
        VS_ERR_OPEN_FORMAT          = VS_ERR_OPEN_INUSE + 1,
        VS_ERR_CLOSE                = VS_ERR_OPEN_FORMAT + 1,
        VS_ERR_START                = VS_ERR_CLOSE + 1,
        VS_ERR_STOP                 = VS_ERR_START + 1,
        VS_ERR_STATE                = VS_ERR_STOP + 1,
        VS_ERR_SET                  = VS_ERR_STATE + 1,
        VS_ERR_SET_SCALER           = VS_ERR_SET + 1,
        VS_ERR_SET_CLIPPER          = VS_ERR_SET_SCALER + 1,
        VS_ERR_NO_MEMORY            = VS_ERR_SET_CLIPPER + 1,
        VS_ERR_INTERNAL             = VS_ERR_NO_MEMORY + 1,
        VS_ERR_NOT_SUPPORTED        = VS_ERR_INTERNAL + 1
    }   VSReturn;
```

```
enum VideoFormat {
        VF_RGB32_A888 = 0, // 4 bytes per pixel, Alpha, R8, G8, B8
        VF_RGB24_888,          // 3 bytes per pixel, R8, G8, B8
        VF_RGB16_565,          // 2 bytes per pixel, R5, G6, B5
        VF_RGB15_A555,         // 2 bytes per pixel, Alpha, R5, G5, B5
        VF_RGB15_55A5,         // 2 bytes per pixel, R5, G5, Alpha, B5
        VF_RGB8_332,           // 1 byte per pixel, R3, G3, B2
        VF_RGB16_DITHER,       // 2 bytes per pixel, R5, G6, B5 dithered
        VF_RGB15_DITHER,       // 2 bytes per pixel, Alpha, R5, G5, B5 dithered
        VF_RGB15_DITHER_1,     // 2 bytes per pixel, R5, G5, Alpha, B5 dithered
        VF_RGB8_DITHER,        // 1 byte per pixel, R3, G3, B2 dithered
        VF_YUV444,             // 3 bytes per pixel, Y, V, U
        VF_YUV422,             // 4 bytes per 2 pixels, Y1, V0, Y0, U0
        VF_YUV411,             // 12 bytes per 8 pixels
        VF_YUV8,               // 1 byte per pixels, Y4, U2, V2
        VF_YUV8_GRAY,          // 1 byte per pixels, Y8
        VF_YUV444_PLANAR,      // YUV 444 planar
        VF_YUV422_PLANAR,      // YUV 422 planar
        VF_MPEG420_PLANAR,     // MPEG 422 planar
        VF_YUV9_PLANAR,        // YUV 9 index planar
        VF_Y2,                 // 4 bytes per 16 pixels, 2 bits of Y
        VF_Y1,                 // 4 bytes per 32 pixels, 1 bit of Y
        VF_YUV2,               // 4 bytes per 8 pixels, 2 bits Y,U,V
        VF_YUV1,               // 4 bytes per 16 pixels, 1 bit Y,U,V
        VF_YUV8_DITHER,        // 1 byte per pixels, Y4, U2, V2 dithered
        VF_YUY2                // 4 bytes per 2 pixels, Y0, U0, Y1, V0
    } VideoFormat;


    {   DEBI_NO_ERROR = 0,             // no error
        CONFIGURE_DENC_ERROR,          // error configuring DENC 7185
        CONFIGURE_DMSD_ERROR,          // error configuring DMSD 7111
        DEBI_FILE_OPEN_ERROR,          // error opening file
        DEBI_ISR_ERROR,                // error registering interrupt service
                                           routine
        DEBI_DMA_TIMEOUT_ERROR,        // error sending data to decoder
```

```
    DEBI_NULL_POINTER,              // null pointer passed to function
    DEBI_MEMORY_ERROR,              // decoder did not accept command
    DEBI_LOWER_VOLUME_LIMIT,        // lower audio volume limit has been
                                        reached
    DEBI_UPPER_VOLUME_LIMIT,        // upper audio volume limit has been
                                        reached
    DEBI_TIMEOUT_ERROR,             // transfer timed out
    DEBI_ERROR,                     // general error encountered
    DSP_COMMAND_ERROR,              //
    DEBI_OUT_OF_RANGE,              // value out of range
    DEBI_COMMAND_ERROR,             // invalid command for current decoder
                                        state
    DSP_COMPLETION_ERROR,           // completion code from DSP not found
    DEBI_MICROCODE_FILE_ERROR,      // open of DSP microcode file failed.
    DSP_WRAP_DATA_ERROR,            // mailbox wrap around test returned bad
                                        data
    DSP_WRAP_TIMEOUT_ERROR,         // mailbox wrap around test timed out
    DEBI_TIMECODE_ERROR,            // error reading SMPTE time code
    DEBI_STARTATFRM_ERROR
} DEBIReturn;

enum VideoPort                      //Video port type
    {
    PORT_D1A = 1,                   // 8 bit port A of type D1
    PORT_D1B,                       // 8 bit port B of type D1
    PORT_DMSD2_AB,                  // 16 bit DMSD2 port Y on A, UV on B
    PORT_DMSD2_BA                   // 16 bit DMSD2 port Y on B, UV on A
    };

enum VelaVideoSignal    // Video signals control port
    {
    VS_KEEP_AS_IS = 0,              // Keep selection as is
    VS_D1A,                         // Video signals from port D1A(Live)
    VS_D1B,                         // Video signals from port D1B(MPEG)
    };
```

```
enum VelaFieldDetect     // Sync and field control
   {
   SFC_KEEP_AS_IS,              // Keep selection as is
   SFC_HR_VR_FDIRECT,           // H-rising, V-rising, F-direct
   SFC_HR_VF_FDIRECT,           // H-rising, V-falling, F-direct
   SFC_HR_VR_FFORCE,            // H-rising, V-rising, F-force toggle
   SFC_HR_VF_FFORCE,            // H-rising, V-falling, F-force toggle
   SFC_HR_VR_FFREE,             // H-rising, V-rising, F-free toggle
   SFC_HR_VF_FFREE,             // H-rising, V-falling, F-free toggle
   SFC_HR_VRF_FDIRECT,          // H-rising, V-rising and falling, F-direct
   SFC_ENCODED                  // H, V and F derived from SAV and EAV
   };

typedef struct               // Field control information
   {
   VelaVideoSignal nFDSource;    // Field detection signals source
   VelaFieldDetect nFDType;       // Field detection type
      } tVelaFieldCtl;
typedef struct               // Video port information
   {
   VideoPort         nPort;              // Video port type
   int               nFrameRate;         // Video frame rate
   tVelaFieldCtl     FieldCtl;           // Sync and field control info.
   VRECT             VAcqRect;           // Acquisition rectangle
   } tVelaVideoPort;

typedef enum VideoType {
      VIDEO_ODD_FIELD= 0,
      VIDEO_EVEN_FIELD= VIDEO_ODD_FIELD + 1,
      VIDEO_INTERLACE= VIDEO_EVEN_FIELD + 1,
      VIDEO_INTER_OE= VIDEO_INTERLACE + 1
   }  VideoType;

typedef enum VideoFormat {
      VF_RGB32_A888= 0,
      VF_RGB24_888= VF_RGB32_A888 + 1,
      VF_RGB16_565= VF_RGB24_888 + 1,
```

```
        VF_RGB15_A555= VF_RGB16_565 + 1,
        VF_RGB15_55A5= VF_RGB15_A555 + 1,
        VF_RGB8_332= VF_RGB15_55A5 + 1,
        VF_RGB16_DITHER= VF_RGB8_332 + 1,
        VF_RGB15_DITHER= VF_RGB16_DITHER + 1,
        VF_RGB15_DITHER_1= VF_RGB15_DITHER + 1,
        VF_RGB8_DITHER= VF_RGB15_DITHER_1 + 1,
        VF_YUV444= VF_RGB8_DITHER + 1,
        VF_YUV422= VF_YUV444 + 1,
        VF_YUV411= VF_YUV422 + 1,
        VF_YUV8= VF_YUV411 + 1,
        VF_YUV8_GRAY= VF_YUV8 + 1,
        VF_YUV444_PLANAR= VF_YUV8_GRAY + 1,
        VF_YUV422_PLANAR= VF_YUV444_PLANAR + 1,
        VF_MPEG420_PLANAR= VF_YUV422_PLANAR + 1,
        VF_YUV9_PLANAR= VF_MPEG420_PLANAR + 1,
        VF_Y2= VF_YUV9_PLANAR + 1,
        VF_Y1= VF_Y2 + 1,
        VF_YUV2= VF_Y1 + 1,
        VF_YUV1= VF_YUV2 + 1,
        VF_YUV8_DITHER= VF_YUV1 + 1,
        VF_YUY2= VF_YUV8_DITHER + 1
    }   VideoFormat;

typedef struct              // PCI port information
    {
    VideoFormat  nFormat;           // Video format
    VideoType    nStreamType;       // Video stream type - O, E or I fields
    int          nPitch;            // # of bytes between starting pixel of two
                                       lines
    BOOL         bVanityMode;       // Vanity mode-True to enable
    BOOL         bUseMMU;           // Physical address is the page table address
    DWORD        dwAddr;            // Physical memory base address
    int          nStreamRate;       // Stream frame rate
    tVSIsrFP     pCallBack;         // Ptr to call back-capture to system memory
```

```
void        *pCallBackParam;    // Ptr to callback function's parameter
VRECT       VRect;              // Video window rectangle
} tVelaPciPort;
```

Function * tVSIsrFP

```
// ---------- Function * tVSIsrFP ----------
typedef void    // VS ISR callback – no return value
   (*tVSIsrFP)(             // Callback function pointer
               void *,      // Ptr to callback function's parameter
               DWORD,       // Ptr to physical memory buffer
               DWORD);      // Data size in bytes
// ----------- tVelaPciPort ----------------
```

typedef enum

```
{
FrameStart = 0,   //Specifies that a frame will be used to start playback
TimeCodeStart = 1,   //Specifies that a time code will be used to start playback
} StartType;
```

typedef enum

```
{
FrameEnd = 0,   //Specifies that a frame will be used to end playback
TimeCodeEnd = 1,   //Specifies that a time code will be used to end playback
FramesPlay = 2,   //Specifies that a duration of frames will be used to end playback
} EndType;
```

Part Three

Appendices and Index

Appendix A

Specifications

Vela CineView Pro Family of Decoders

CineView Pro XL Decoder

General

- Supports decoding of the following image resolutions:

NTSC	PAL
352×240	352×288
352×480	352×576
480×480	480×576
544×480	544×576
640×480	704×576
704×480	720×576
720×480	720×608 (4:2:2)
720×512 (4:2:2)	

- Supported stream formats:
  - MPEG-1 Elementary streams
  - MPEG-1 System streams
  - MPEG-2 Elementary streams
  - MPEG-2 Packetized Elementary streams (PES)
  - MPEG-2 Program streams
  - MPEG-2 Transport streams
- Supports MPEG data rates up to 50Mbps (system dependant)
- Reconstruction of I, P, and B frames
- Supports MPEG-1 video compression encapsulated in MPEG-2 Program and Transport streams
- Built-in VGA video display support. VGA display screen brightness, contrast, and saturation adjustments accessible through playback application
- LTC output available
- Simultaneous digital AES/EBU and analog stereo audio outputs.
- Digital/analog stereo audio playback via a 15-pin high density connector.
- Supports MPEG audio layers 1 and 2 at 32kHz, 44.1kHz, and 48kHz.
- Audio output volume level and mute function accessible through playback application and SDK.
- Supports MPEG-2 audio PES.

Digital Video

- SMPTE 259M-C; 270Mbps; 525/625 line component serial digital (SDI) output via BNC connector
- Format fully Compliant With SMPTE 259M
- Amplitude: 800mv ± 10%
- DC Offset: 0.0v ± 0.5v
- Rise and fall times: No less than 0.4ns; no greater than 1.5ns
- Overshoot: Less then 10% of amplitude
- Timing Jitter: 0.2 µI P-P
- Bit Rate: 270 Mbps
- Output Characteristics: 75-ohm, NRZI Serial Data

Analog Video

- Genlock input via BNC connector
- Video output via BNC connector
- Output Level: 1V P-P
- Frequency Response
  - NTSC: 0 to 4.0 MHz ± 1dB
  - PAL: 5.8 MHz +1, -7dB
- Differential Gain: ±2%
- Differential Phase ±1.3°
- SN Ratio: >65dB, unweighted
- Color Burst Amplitude
  - NTSC: 40 IRE ±1 IRE
  - PAL: 43 IRE ±1 IRE
- Luminance Level Accuracy: ±1 IRE
- Chrominance Level Accuracy: ±1 IRE (All Colors)
- Sync Amplitude
  - NTSC: 40 IRE ±1 IRE
  - PAL: 43 IRE ±1 IRE

Digital Audio

- 15-pin high-density D-sub connector
- AES3-1992; 110 ohms, balanced output; 75 ohms, unbalanced output
- SMPTE 337M AES/EBU compressed bitstream audio output
- Impedance (jumper selectable)
  - Single Ended Coaxial: 75 Ohm
  - Differential: 110 Ohm

- Jitter:
  - 75 Ohm: ±10ns
  - 110 Ohm: ±10ns
- Fall Time:
  - 5ns <30ns (110-ohm differential termination)
  - 30ns <44ns (75-ohm termination)
- Rise Time
  - 5ns <30ns (110-ohm differential termination)
  - 30ns <44ns (75-ohm termination)
- Output voltage
  - 75-ohm: 1.0v P-P ±10%
  - 110-ohm: 2.2v P-P ±10%

Analog Audio

- 15-pin high-density D-sub connector
- Dual two-channel decoders
- Frequency Response: 20Hz to 20kHz ±0.3dB
- Line Output Level: 0 VU = +4dBm (analog output)
- S/N Ratio: >90 dB Minimum
- THD: <0.03% @ 1kHz at +4dbm output level
- Stereo Separation: > 80 dB

Note: Audio specifications apply to decoder only. PC sound card or external audio system may degrade overall audio performance.

External Connectors

- Standard PCI edge connector for mounting
- BNC SDI video output & 4 channels of embedded audio
- BNC composite video output (PAL/NTSC)
- BNC genlock input
- DB-15 high density D-sub audio output:
  - Digital AES/EBU: 110-ohm balanced / 75 ohm unbalanced (jumper selectable)
  - Balanced analog audio
- DB-9 D-sub connector for LTC output on optional LTC bracket

Physical Size

- Full-length PCI form factor

Power Requirements

- +3.3VDC ±5% @ 3.1A (typ); 3.5A (typ) with audio embedder
- +5VDC ±5% @ 1.5A (typ)

- +12VDC ±5% @ 160ma (typ)
- - 12VDC ±5% @ 50ma (typ)

Operating Environment

- Temperature:
  - Operating: 41° to 104°F (5° to 40°C).
  - Non-operating: –40°F to 149°F (–40°C to 65°C).
- Humidity (Non-condensing):
  - Operating: 10% to 90%.
  - Non-operating: 5% to 95%, packaged.
- Maximum wet bulb humidity:
  - Operating: 80.6°F (27°C).
  - Non-operating: Non-condensing.
- Air flow @77°F (25°C):
  - Operating: TBD.
  - Non-operating: N/A.
- Altitude:
  - Operating: 0 to 2.24 miles (0 to 3.6 kilometers).
  - Non-operating: 0 to 2.24 miles (0 to 3.6 kilometers).

CineView Pro/Pro LE Decoders

Video

- Supports decoding of the following image resolutions:

NTSC	PAL
352×240	352×288
352×480	352×576
480×480	480×576
544×480	544×576
640×480	704×576
704×480	720×576
720×480	720×608 (4:2:2)
720×512 (4:2:2)	

- Supported stream formats:
  - MPEG-1 Elementary streams
  - MPEG-1 System streams
  - MPEG-2 Elementary streams

MPEG-2 Packetized Elementary streams (PES)

MPEG-2 Program streams

MPEG-2 Transport streams

- Supports MPEG data rates up to 50Mbps (system dependant)
- Reconstruction of I, P, and B frames
- Supports MPEG-1 video compression encapsulated in MPEG-2 Program and Transport streams
- Built-in VGA video display support. VGA display screen brightness, contrast, and saturation adjustments accessible through playback application
- External video output:

  SMPTE 259M-C; 270Mbps; 525/625 line component serial digital (SDI) output via BNC connector (CineView Pro only).

  Analog video out via BNC connector.
- Genlock input via BNC connector
- Digital Video Figures (CineView Pro):

  Return loss: 18dB to clock frequency.

  Signal level: 800mV ±10%.

  DC offset: 0V ±0.5V.

  Rise and fall time: 400–1500pS (20% – 80%).
- Analog Video Figures:

  Output level: 1V p-p.

  Frequency response: NTSC: 0– 4.0 MHz ±1dB.
  PAL: 5.8 MHz +1dB, -.7dB.

  Differential gain: 2%.

  Differential phase: 1.3°.

  S/N: >65dB, unweighted.

  Color burst amplitude: NTSC: 40 IRE ±1 IRE.
  PAL: 43 IRE ±1 IRE.

  Luminance level accuracy: ±1 IRE.

  Chrominance Level Accuracy: ±1 IRE.

  Sync Amplitude: NTSC: 40 IRE ±1 IRE.
  PAL: 43 IRE ±1 IRE.

Audio

Note: Audio specifications apply to decoders only. PC sound card or external audio system may degrade overall audio performance.

- Selectable digital/analog stereo audio playback via a 15-pin high density connector.
- Digital audio formats include AES/EBU (default) and S/P DIF. (Digital formats apply to CineView Pro only.
- Supports MPEG audio layers 1 and 2 at 32kHz, 44.1kHz, and 48kHz.

- Audio output volume level and mute function accessible through playback application and SDK.
- Supports MPEG-2 audio PES.
- Digital Audio Figures: (CineView Pro only)

  Connector: 15-pin high-density D-sub.

  Standard: AES3-1992; 110 ohms, balanced output; 75 ohms, unbalanced output.

  Simultaneous digital AES/EBU and analog stereo audio outputs.
- Analog Audio Figures:

  Connector: 15-pin high-density D-sub.

  Output level: +4dBm = 0Vu.

  S/N: 88dB minimum.

  Frequency response: 20Hz to 20kHz, ±0.2dB.

  Line output level: 0Vu = +4dBm.

  Total Harmonic Distortion: THD +N <0.05% (1kHz at +4dBm output level).

  Stereo separation: > 80dB.

External Connectors

- Standard PCI connector
- DB-15 high density D-sub audio output:

  Digital AES/EBU — 110 ohm balanced/75 ohm unbalanced (CineView Pro only).

  Balanced analog audio.
- BNC component SDI video output (CineView Pro only)
- BNC composite video output (PAL/NTSC)
- BNC genlock input

Physical

- Dimensions

  Length: 6.9” (175mm) (half-length PCI form factor).

  Height: 4.2” (107mm).

Power Requirements

- +5VDC, 1.7A. ±5%
- +12VDC, 0.15A. ±5%
- –12VDC, 0.03A. ±5%

Operating Environment

- Temperature:

  Operating: 41° to 104°F (5° to 40°C).

Non-operating: –40°F to 149°F (–40°C to 65°C).

- Humidity (Non-condensing):
  Operating: 10% to 90%.
  Non-operating: 5% to 95%, packaged.
- Maximum wet bulb humidity:
  Operating: 80.6°F (27°C).
  Non-operating: Non-condensing.
- Air flow @77°F (25°C):
  Operating: TBD.
  Non-operating: N/A.
- Altitude:
  Operating: 0 to 2.24 miles (0 to 3.6 kilometers).
  Non-operating: 0 to 2.24 miles (0 to 3.6 kilometers).

All specifications subject to change without notice.

Appendix B

Troubleshooting

Symptom	Possible Solutions
No Video	• Make sure that the VGA driver enables linear addressing to the frame buffer. • Make sure that the CineView Pro series decoder board is mounted in a PCI bus master slot. (Check the motherboard documentation to determine bus master slots.) • Check if PCI BIOS has enabled the bus master enable bit within the PCI configuration space. • If in composite output mode, close application and try starting it again to ensure that the composite output circuitry is initialized correctly. • Not a valid MPEG stream or a video elementary stream. (The decoder currently handles MPEG-1 and MPEG-2 system, transport, and program streams, also video and audio packetized elementary streams.) • Make sure the VGA controller is in High Bit 16 or High Bit 24. • If application is halted, try to close the application and restart it.
Video Artifacts	• *Slow Video Update:* Artifacts in the video can be due to insufficient interrupt latency timing on the PCI bus. The recommended latency is 60-80. • *Video Too Slow:* The video input source is too slow. This could happen when the MPEG file is being played off a CD-ROM that is too slow. In this case, move the MPEG stream to hard drive. • *Video Macroblocking* and application lockup could be a result of the board locking up as a result of a problem with the stream. Close all applications and reboot the computer system.
Audio Artifacts	• Audio stream is not a packetized elementary stream.

Table B-1. Troubleshooting Guide

If this short troubleshooting guide cannot help you resolve problems related to operation of the CineView Pro series decoder, please contact the Vela Training and Support staff as described on page 32 of Chapter 1.

Index

P

R

S

T

U

V

W

X